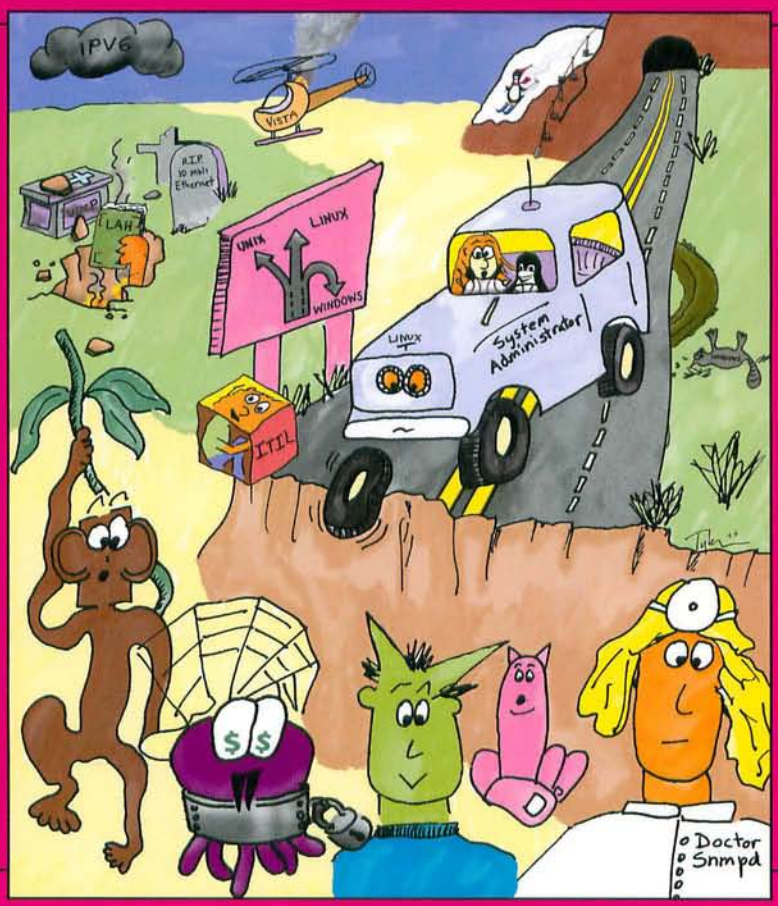
Linux之父 Linus Torvalds倾力推荐

Linux 系统管理技术手册（第二版）

LINUX ADMINISTRATION HANDBOOK



Evi Nemeth
［美］ Garth Snyder 著
Trent R. Hein
张 辉 译

人民邮电出版社
POSTS & TELECOM PRESS

Linux
系统管理技术手册
（第二版）
LINUX ADMINISTRATION HANDBOOK

〔美〕 Evi Nemeth
Garth Snyder 著
Trent R. Hein
张 辉 译

人 民 邮 电 出 版 社
北 京

www.TopSage.com

图书在版编目（CIP）数据

Linux 系统管理技术手册；第 2 版 /（美）奈米斯（Nemeth, E.），（美）斯奈德（Snyder, G.），（美）海因（Hein, T.R.）著；张辉英译中．—北京：人民邮电出版社，2008.5
ISBN 978-7-115-17514-4

Ⅰ．L… Ⅱ．①奈…②斯…③海…④张… Ⅲ．Linux 操作系统—技术手册 Ⅳ．TP316.89-62

中国版本图书馆 CIP 数据核字（2008）第 006046 号

版 权 声 明

Authorized translation from the English language edition, entitled LINUX ADMINISTRAITON HANDBOOK, 2nd Edition, 0131480049 by Nemeth, Evi; Snyder, Garth, published by Pearson Education, Inc, publishing as Prentice Hall, Copyright © 2007 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD.,and POSTS & TELECOMMUNICATIONS PRESS Copyright © 2008.

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签。无标签者不得销售。

Linux 系统管理技术手册（第二版）

◆ 著 [美] Evi Nemeth Garth Snyder Trent R.Hein
译 张 辉
责任编辑 李 际

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：49.5
字数：1 436 千字 2008 年 5 月第 2 版
印数：9 501 – 13 500 册 2008 年 5 月北京第 1 次印刷
著作权合同登记号 图字：01-2007-0863 号

ISBN 978-7-115- 17514-4/TP

定价：99.00 元

读者服务热线：(010)67132705 印装质量热线：(010)67129223
反盗版热线：(010)67171154

www.TopSage.com

内容提要

本书延续了本书前一版以及《UNIX 系统管理技术手册》的讲解风格，以当前主流的 5 种 Linux 发行版本（Red Hat ES、SUSE、Debian、Fedora Core 和 Ubuntu）为例，把 Linux 系统管理技术分为 3 个方面分别介绍。第 1 部分全面介绍了运行单机 Linux 系统涉及的各种管理知识和技术，如系统引导和关机、进程控制、文件系统管理、用户管理、设备管理、系统备份、软件配置以及 cron 和系统日志的管理使用等。第 2 部分从详细讲解 TCP/IP 协议基本原理开始，深入讨论了网络的两大基本应用——域名系统和路由技术，然后逐章讲解 Linux 上的各种 Internet 关键应用，如电子邮件、NFS、文件共享、Web 托管和 Internet 服务，在这部分里还有专门的章节介绍网络硬件、网络管理与调试以及系统安全。第 3 部分包括了多种不容忽视的重要主题：X 窗口系统、打印系统、系统维护与环境、性能分析、与 Windows 系统的协作、串行设备、操作系统驱动程序和内核、系统守护进程以及政策与行政管理方面的知识等。

本书的几位作者是分别来自学术界、企业界以及职业培训领域的 Linux/UNIX 系统管理专家，这使得本书从第一版开始，即成为全面、深入而且颇富实用性的 Linux 系统管理权威参考书。本书适合于从 Linux 初学者到具有丰富经验的 Linux 专业技术人员使用。

www.TopSage.com

大家网
TopSage.com

关于作者



Evi Nemeth 已经从科罗拉多大学（University of Colorado）计算机科学系教师的岗位上退休了，但是她仍然在参与 CAIDA 的网络研究工作，CAIDA 是圣地亚哥超级计算中心（San Diego Supercomputer Center）的 Internet 数据分析协作组织（Cooperative Association for Internet Data Analysis）。她现在正坐在自己的新玩具——12 米长的帆船 Wonderland 号——上遨游加勒比海呢。



Garth Snyder 曾经在 NeXT 和 Sun 公司工作过，他从斯沃索莫学院（Swarthmore College）获得了电机工程专业的学位，并且在罗彻斯特大学（University of Rochester）取得 MD 和 MBA 学位。



Trent R. Hein 是 Applied Trust Engineering 公司的创办人之一，这是一家提供网络基础设施的安全和性能咨询服务的公司。Trent 从科罗拉多大学（University of Colorado）获得了计算机科学专业学士学位。

www.TopSage.com

关于供稿人

Lynda McGinley 在教育领域从事系统和网络管理工作有 20 年了，她曾经是《UNIX 系统管理技术手册》（第三版）（中文版已由人民邮电出版社出版）的供稿人之一。现如今，她在美国大学大气研究联盟（University Corporation for Atmospheric Research）担任系统安全工程师。Lynda 相信，自己在科罗拉多大学为 Evi 做了数年系统管理员的经历会让大家牢牢记住她。

Ben Whaley 是 Applied Trust Engineering 公司的一名资深工程师。他获得了科罗拉多大学计算机科学专业的学位，并且还是 Red Hat 认证工程师。Ben 不但从事 Perl 和 PHP 编程，而且在启蒙课程中担任教师。

在 20 世纪 90 年代后期，Adam Boggs 一直在科罗拉多大学本科生实验室做第一线的学生系统管理员。前几年他在从事内核开发工作，为 Solaris 和 Linux 系统开发文件系统和存储管理。目前，他在为用于大气建模的超级计算机做网络、存储和集群管理工作。

Jeffrey S. Haemer 从 1983 年起就一直在从事与 Linux 和商业版的 UNIX 相关的工作，他始终坚持不用 Microsoft Word 软件。他前前后后在打印机制造公司 QMS 里工作过 11 年。Evi 曾经教 Jeff 开车。他最喜爱的口头禅是："A man, a plan, a canal, Suez"。

Tobi Oetiker 是电气工程师，工作干的却是系统管理员。他从 1994 年起就开始做系统管理工作。他在苏黎世理工（ETH Zurich）工作了多年，为那里的师生建立了一个功能丰富的计算环境，同时还编写了几种流行的开放源代码软件，例如 MRTG、RRDtool 和 SmokePing，现在他在 Oetiker+Partner AG 公司工作，这是一家提供 IT 和咨询服务的公司。他已经结婚，生活在瑞士。

Fritz Zaucker 学的是物理学。他先前从事了几年气候变化方面的研究，之后领导苏黎世理工（ETH Zurich）信息技术和电气工程系的 IT 支持小组（IT Support Group，ISG.EE）长达 10 年之久。他和一群志同道合的同事们一起，把 ISG.EE 变成了一个专业机构，主要面向高效系统管理、可靠 IT 基础设施以及满足客户需求。现在他加入了

www.TopSage.com

Oetiker+ Partner AG 公司，和他的一些朋友和同事一起工作。Oetiker+ Partner AG 公司是一家 IT 和管理咨询公司，同时还提供 IT 服务。Fritz 同 Evi 及其侄女一起在地中海驾艇畅游，诚心邀请读者朋友们也来 Wonderland 号上和她一起享受一下。

Scott Seidel 是 Applied Trust Engineering 公司的一位资深工程师，他在该公司专门研究企业安全和性能管理。他从位于玻尔得（Boulder）的科罗拉多大学获得工商管理学位，研究重点在信息系统和金融方面。Scott 感兴趣的研究领域包括系统监测和服务器虚拟化。在管理 Linux 系统的工作之余，Scott 喜爱在厨房里自酿新酒。

Bryan Buus 在 Adeptive Software 公司担任负责工程的副总裁，这是一家承接定制软件开发的公司。在网络公司（.COM）热潮之前，Bryan 是 XOR 公司负责开发的副总裁，他领导该公司的 Web 和 Internet 服务部门。Bryan 获得波士顿大学计算机科学学士和硕士学位。

Ned McClain 是 Applied Trust Engineering 公司的创办人之一，也是该公司的 CTO。在美国的 USENIX 和亚洲的 APRICOT 这样的技术大会上，他主讲各种涉及系统管理和安全的专题。Ned 从美国康奈尔大学工学院获得计算机科学专业的学位，他还是一位 CISSP（编号 39389）。

David Schweikert 在瑞士的苏黎世理工（ETH Zurich）做系统管理员，他在那里主要负责电气工程系的电子邮件系统。他是开放源代码项目 Mailgraph（绘制邮件统计图的工具）和 Postgrey（用于 Postfix 的一种 greylisting[灰名单]技术实现）的开发者。David 还对数据库和大型站点的系统管理有兴趣。

www.TopSage.com

第一版序

能预先看到这本书，我感觉到异常的兴奋，因为这是 *UNIX System Administration Handbook*（USAH）一书专为 Linux 出版的最新版本。*USAH* 第三版已经包含了 Red Hat Linux 的内容，但当时它仅仅是 4 种不同的 UNIX 变体之一。本书的这一版介绍的则全部是几种不同的 Linux 发行版本，原先与 Linux 无关的素材大多被略去不再赘述了。我自己也很好奇，想看看它究竟和以前有多少不同之处。

太多了，它涵盖的内容真是太多了。Linux 各个发行版本都是从开放源代码软件的共同资源中提炼出来的，所以比起 UNIX 的其他版本来说，它们彼此之间要相像得多。因此本书内容的针对性显得愈发突出。几位作者不但阐述了您的系统运行“可能”采取的各种不同方式，而且现在还会准确地告诉您，它是如何“那样”运行的。

与此同时，本书仍然全面介绍了 UNIX 种类繁多的丰富软件。现在，世界上的流行软件几乎全都能在 Linux 上运行，而 Linux 的站点却发现它们所面对的威胁越来越少。随着像 IBM、Oracle 和 SGI 这样的业界巨人展开双臂热情地将 Linux 拥入怀中，它正在迅速成为世界的标准，人们会拿别的 UNIX 版本和这个标准做比较（而且还不一定能比得上！）。

正如本书所体现出来的那样，Linux 系统和与之对应的那些专有 UNIX 一样功能强大、一样安全，也一样可靠。幸亏有了数以千计的 Linux 开发人员无时无刻的努力，才让 Linux 比以前任何时候都准备得更充分，更适合投入到现实世界的各条“战线”上。本书的几位作者十分熟悉“地形”，所以我很高兴地把您交到他们最有能力的手中。尽情享受阅读带来的乐趣吧！

Linus Torvalds——Linux 之父

2002 年 4 月

www.TopSage.com

致　谢

许多人都以这样或者那样的方式为本书的编写工作提供了帮助，他们的帮助包括从技术评审或者习题建议一直到精神上全面支持的各个方面。我们要特别感谢下面这些人士，感谢他们一直给予我们的鼓励：

Bo Connell	Jon Corbet	Jim Lane
Sam Leffler	Cricket Liu	Derek Martin
Laszlo Nemeth	Eric Robinson	Sam Stoller
Paul Vixie	Aaron Weber	Greg Woods

我们不但要感谢 Prentice Hall 公司负责本书的编辑 Catherine Nolan 和 Mary Franz，而且还要褒奖她们能够成功地和我们这几个有点儿不靠谱的作者以及一群“配角人物”打好交道，后者有时候似乎就是多达成千上万的供稿人。

作为版面文字编辑，Mary Lou Nohr 再次做出了异常优秀的工作成绩。她做到了集补救高手和润色大师的角色于一身。我们要说，我们今后将非常高兴同她再度合作。

Mark G • Sobell 在索引上面肯定付出了细致和耐心的工作。我们对结果非常满意，多谢他的帮助。

最后，Evi 要向加勒比海沿岸海滩上的众多酒吧和咖啡屋表示歉意，同时也要谢谢它们。Evi 把她的帆船停泊在信号最强的地方，偷偷地使用了它们的免费无线网络接入服务。当 Evi 坐在船上把身边的天堂美景抛到一边，忙于处理本书各个章节的时候，她发誓说这将是她写的最后一本。不过，她说的这话谁会当真呢？

www.TopSage.com

大家网
TopSage.com

前　言

当我们在大约 5 年前编写本书第一版的时候，Linux 才刚刚开始在企业级的应用领域中崭露头角。那时候，我们希望本书第一版会有助于传播这样的信息，即 Linux 已经是一种最高档次的操作系统，具备了同 Sun、HP 和 IBM 公司的系统产品相媲美的实力。

现如今 Linux 已经是 IBM 提供的产品了。IBM 在 2004 年宣布其全线服务器均支持 Linux，对于任何还在等待明确无误的信号来表明 Linux 这汪池水对于企业级游泳者已经没有危险的人们来说，这个消息非常振奋人心。没有人曾经因为购买了 IBM 的产品而丢掉工作，采用 Linux 基本上也是一项同样保险的提议。[1]

我们着手撰写本书的目的，就是让它成为专业 Linux 系统管理员的最好朋友。在任何合适的地方，我们都会重新编写我们以前写过的深受读者欢迎的书—《UNIX 系统管理技术手册》（已由人民邮电出版社出版）—所验证过的概念和资料。我们加上了大量专门针对 Linux 的素材，并且把其余内容根据最新情况做了调整，但是本书的覆盖面仍然保持和原来的类似。我们希望您能够认可这一点，即这样做的结果会给读者送来一本高质量的 Linux 系统管理工作指南，它得益于过去几个版本的经验。

Linux 系统管理方面也有不少别的书籍，但是其中没有一本能在深度和广度上，提供在现实的商业环境下高效地使用 Linux 所必须掌握的资料。下面是本书有别于他人的突出特色。

- 我们采取了一种结合实践讲述问题的方式。我们的目的不是复述系统手册上的内容，而是总结我们在系统管理工作中积累起来的经验。这本书包含了许多在现实中向困难宣战的故事，也给出了大量注重实践的建议。
- 这本书不是讲如何在家里、在车库里或者在 PDA 上运行 Linux。我们介绍的是如何在实际工作的环境下，比如商业公司、政府机关以及大学里使用 Linux。
- 我们详细地讲解了 Linux 的连网技术。这是系统管理工作中最为困难的方面，也是我们认为自己可以向读者提供最多帮助的领域。
- 我们没有过于简化材料。我们的例子都反映出了实际中的真实情况，并没有掩盖它们所有的繁冗和复杂。在大多数情况下，这些例子都是直接取自实际工作的系统。
- 我们介绍了 5 种主流的 Linux 发行版本。

我们举例的发行版本

和许多操作系统一样，Linux 也在成长过程中出现了几个不同方向的分支。虽

[1] 至少对于服务器而言是这样。现在的竞争集中在台式机上，微软的 Windows 仍然在这个领域保持着几乎垄断的地位。Linux 和 Windows 竞争的结果尚难预料。到目前为止，Windows 提供的用户界面仍然更精美一些。但是要想战胜“免费”的 Linux 可绝非易事。

www.TopSage.com

然内核的开发仍然保持高度的集中，但是打包和发布完整 Linux 操作系统的工作则由各种不同的组织来完成，每家都有自己的套路。

我们详细介绍了 5 种 Linux 发行版本：

- Red Hat® Enterprise Linux® 4.3 ES；
- Fedora™ Core 5；
- SUSE® Linux Enterprise 10.2；
- Debian® GNU/Linux 3.2 “Etch”（2006 年 9 月的测试发布）；
- Ubuntu® 6.06 “Dapper Drake”。

我们之所以选择这几种发行版本，不仅因为它们最流行，而且因为它们是整个 Linux 界的代表。不过，本书里的许多内容也都能应用到其他主流的发行版本上。

对于我们所讨论的每个主题，我们都提供了有关每种发行版本的详细信息。针对某种特定操作系统的注释，则用发行版本的徽标标出。

本书的组织

本书分为 3 大部分：基本管理技术、网络管理技术和其他管理技术。

基本管理技术部分从系统管理员的角度全面介绍 Linux。其中的章节涉及运行单机 Linux 系统所需要的大部分知识和技术。

网络管理技术部分描述了 Linux 系统使用的各种协议，介绍了用来安装、扩展和维护网络所使用的各种技术。在这个部分中还介绍了高层网络软件。各章的专题内容包括域名系统、网络文件系统、路由技术、**sendmail** 和网络管理。

其他管理技术部分包括各种各样的补充信息。其中有些章节讨论了一些可选的软件包，例如 Linux 的打印系统。其他一些章节就各种主题——从硬件维护到 Linux 安装的执行策略——提供了若干建议。

每一章的后面还有一组练习题。我们用星号标出了我们估计要完成这道题所需付出的努力，“努力”则体现在题目的难度和需要花费的时间两方面。

题目有 4 级：

没有星号简单题目，应该很容易就能做出来；

★ 比较难或者要花比较长时间的题目，可能要求做实验；

★★ 最困难或者最花时间的题目，要求做实验，并进行深入分析；

★★★★ 作为整个学期的项目（只在个别章节出现）。

有些习题需要有系统上的 root 或者 sudo 权限，有些题目要求得到本地系统管理小组的许可。有这两种要求的时候，习题会予以说明。

我们的供稿人

我们很高兴 Adam Boggs、Bryan Buus 和 Ned McClain 能以供稿人的身份再度参与本书的编写工作。我们还欢迎 Ben Whaley、Tobi Oetiker、Fritz Zaucker、Jeffrey S. Haemer、David Schweikert 和 Scott Seidel 在这一版也成为我们的供稿人和朋友。他们在各个领域的深厚知识已经极大地丰富了本书的内容。我们首先要向 Lynda McGinley 表示感谢，她不但编写了大量文字内容，而且还不知疲倦地组织和协调了供稿人的工作。

我们希望您会喜欢本书，并祝您的系统管理工作好运连连！

Evi Nemeth
Garth Snyder
Trent R. Hein
2006 年 10 月

www.TopSage.com

目　录

第一部分　基本管理技术……1

第1章　从何处入手……2
1.1　读者的知识背景……2
1.2　Linux 同 UNIX 的关系……3
1.3　历史上的 Linux……4
1.4　Linux 的发行版本……4
1.4.1　最好的发行版本是什么……6
1.4.2　特定于发行版本的系统管理工具……7
1.5　表示法和印刷约定……7
1.6　从哪里获得信息……8
1.6.1　手册页的组织……9
1.6.2　man：读取手册页……10
1.6.3　其他的 Linux 信息资源……10
1.7　如何找到和安装软件……11
1.8　系统管理员的基本任务……12
1.8.1　增加、删除以及管理用户账号……12
1.8.2　增删硬件……13
1.8.3　执行备份……13
1.8.4　安装和更新软件……13
1.8.5　监视系统……13
1.8.6　故障诊断……13
1.8.7　维护本地文档……13
1.8.8　时刻警惕系统安全……14
1.8.9　帮助用户……14
1.9　重压下的系统管理员……14
1.10　推荐读物……15
1.11　习题……15

第2章　引导和关机……17
2.1　引导……17
2.1.1　自动引导和手工引导……18
2.1.2　引导过程的步骤……18
2.1.3　初始化内核……18
2.1.4　配置硬件……18
2.1.5　内核线程……19
2.1.6　操作员干预（仅限手工引导）……19

www.TopSage.com

2.1.7 执行启动脚本……20
2.1.8 多用户运行……20
2.2 引导 PC……20
2.3 引导加载程序：LILO 和 GRUB……21
2.3.1 GRUB：全面统一的引导加载程序……21
2.3.2 LILO：传统的 Linux 引导加载程序……22
2.3.3 内核选项……23
2.3.4 PC 上的多重引导……24
2.3.5 GRUB 的多重引导配置……24
2.3.6 LILO 的多重引导配置……25
2.4 引导单用户模式……25
2.4.1 用 GRUB 引导单用户模式……26
2.4.2 用 LILO 引导单用户模式……26
2.5 启动脚本……26
2.5.1 init 和运行级……27
2.5.2 Red Hat 和 Fedora 的启动脚本……29
2.5.3 SUSE 的启动脚本……30
2.5.4 Debian 和 Ubuntu 的启动脚本……32
2.6 重新引导和关机……32
2.6.1 关闭电源……33
2.6.2 shutdown：停止系统的妥善方式……33
2.6.3 halt：关闭系统的更简单方式……33
2.6.4 reboot：快速的重新启动……34
2.6.5 telinit:改变 init 的运行级……34
2.6.6 Poweroff：请 Linux 关闭电源……34
2.7 习题……34
第 3 章 超级用户的权力……36
3.1 文件和进程的归属关系……36
3.2 超级用户……37
3.3 选择 root 的口令……38
3.4 成为 root 用户……39
3.4.1 su：替换用户身份……39
3.4.2 sudo：受限的 su……39
3.5 其他的伪用户……41
3.5.1 bin：系统命令的老属主……42
3.5.2 daemon：无特权系统软件的属主……42
3.5.3 nobody：普通 NFS 用户……42
3.6 习题……42
第 4 章 进程控制……43
4.1 进程的组成部分……43
4.1.1 PID：进程的 ID 号……44
4.1.2 PPID：父 PID……44
4.1.3 UID 和 EUID：真实的和有效的用户 ID……44

www.TopSage.com

4.1.4 GID 和 EGID: 真实的和有效的组 ID ····45
4.1.5 谦让度 ····45
4.1.6 控制终端 ····45
4.2 进程的生命周期 ····45
4.3 信号 ····46
4.4 KILL 和 KILLALL: 发送信号 ····48
4.5 进程的状态 ····49
4.6 nice 和 renice: 影响调度优先级 ····50
4.7 PS: 监视进程 ····50
4.8 TOP: 更好地监视进程 ····52
4.9 /PROC 文件系统 ····53
4.10 STRACE: 追踪信号和系统调用 ····54
4.11 失控进程 ····54
4.12 推荐读物 ····56
4.13 习题 ····56
第 5 章 文件系统 ····57
5.1 路径名称 ····58
5.2 安装和卸载文件系统 ····59
5.3 文件树的组织 ····60
5.4 文件类型 ····62
5.4.1 普通文件 ····63
5.4.2 目录 ····63
5.4.3 字符设备文件和块设备文件 ····64
5.4.4 本地域套接口 ····65
5.4.5 有名管道 ····65
5.4.6 符号链接 ····65
5.5 文件属性 ····66
5.5.1 权限位 ····66
5.5.2 setuid 和 setgid 位 ····66
5.5.3 粘附位 ····67
5.5.4 查看文件属性 ····67
5.5.5 chmod: 改变权限 ····68
5.5.6 chown: 改变归属关系和组 ····69
5.5.7 umask: 分配默认的权限 ····70
5.5.8 额外的标志 ····70
5.6 访问控制列表 ····71
5.6.1 ACL 概述 ····72
5.6.2 默认项 ····74
5.7 习题 ····74
第 6 章 添加新用户 ····76
6.1 /etc/passwd 文件 ····76
6.1.1 登录名 ····77
6.1.2 加密的口令 ····78

www.TopSage.com

6.1.3 UID 号……78
6.1.4 默认的 GID 号……79
6.1.5 GECOS 字段……79
6.1.6 主目录……80
6.1.7 登录 shell……80
6.2 /etc/shadow 文件……80
6.3 /etc/group 文件……82
6.4 添加用户……83
6.4.1 编辑 passwd 和 shadow 文件……84
6.4.2 编辑/etc/group 文件……84
6.4.3 设置初始口令……85
6.4.4 创建用户的主目录……85
6.4.5 复制默认启动文件……85
6.4.6 设置用户的邮件主目录……86
6.4.7 验证新的登录……87
6.4.8 记录用户的状态和联系信息……87
6.5 删除用户……87
6.6 禁止登录……88
6.7 账号管理工具……88
6.8 习题……89
第 7 章 增加硬盘……91
7.1 硬盘接口……91
7.1.1 PATA 接口……92
7.1.2 SATA 接口……93
7.1.3 SCSI 接口……93
7.1.4 SCSI 和 IDE 的比较……96
7.2 硬盘的几何结构……96
7.3 Linux 文件系统……98
7.3.1 ext2fs 和 ext3fs……98
7.3.2 ReiserFS……98
7.3.3 XFS 和 JFS……99
7.4 硬盘安装过程概述……99
7.4.1 连接硬盘……99
7.4.2 格式化硬盘……100
7.4.3 标记硬盘和硬盘分区……100
7.4.4 在硬盘分区内创建文件系统……101
7.4.5 安装文件系统……102
7.4.6 设置自动安装……103
7.4.7 启用交换分区……104
7.5 hdparm：设置 IDE 接口参数……104
7.6 fsck: 检查和修复文件系统……106
7.7 向 Linux 增加硬盘：一步步的指导……108
7.8 高级硬盘管理：RAID 和 LVM……112

www.TopSage.com

7.8.1 Linux 的软 RAID……112
7.8.2 管理逻辑卷……113
7.8.3 配置 LVM 和 RAID 的例子……113
7.8.4 处理故障硬盘……117
7.8.5 重新分配存储空间……118
7.9 安装 USB 设备……119
7.10 习题……120
第 8 章 周期性进程……122
8.1 cron:按时间表执行命令……122
8.2 crontab 文件的格式……123
8.3 crontab 管理……124
8.4 cron 的常见用途……125
8.4.1 清理文件系统……125
8.4.2 配置文件的网络分布……126
8.4.3 循环日志文件……127
8.5 其他的日程安排程序：ANACRON 和 FCRON……127
8.6 习题……127
第 9 章 备份……129
9.1 备份基本原理……130
9.1.1 从一台机器执行所有的转储……130
9.1.2 给备份介质加卷标……130
9.1.3 选择合理的备份间隔……130
9.1.4 仔细选择文件系统……130
9.1.5 在一卷磁带上做日常转储……131
9.1.6 使文件系统小于转储设备……131
9.1.7 异地保存磁带……131
9.1.8 保护备份……132
9.1.9 执行转储期间限制活动……132
9.1.10 检查磁带……132
9.1.11 发掘磁带的寿命……133
9.1.12 为备份而设计数据……133
9.1.13 作最坏的准备……133
9.2 备份设备和介质……133
9.2.1 光盘：CD-R/RW、DVD±R/RW 和 DVD-RAM……134
9.2.2 可移动硬盘（USB 和火线）……135
9.2.3 小型磁带机：8 毫米磁带和 DDS/DAT……135
9.2.4 DLT 和 S-DLT……135
9.2.5 AIT 和 SAIT……135
9.2.6 VXA 和 VXA-X……136
9.2.7 LTO……136
9.2.8 自动选带机、自动换带机以及磁带库……136
9.2.9 硬盘……137
9.2.10 介质类型小结……137

www.TopSage.com

9.2.11 设备选型……137
9.3 用 dump 建立增量备份机制……138
9.3.1 转储文件系统……138
9.3.2 转储顺序……140
9.4 用 restore 从转储中恢复……141
9.4.1 恢复单个文件……141
9.4.2 恢复整个文件系统……142
9.5 为系统升级而执行转储和恢复……143
9.6 使用其他存档程序……144
9.6.1 tar: 给文件打包……144
9.6.2 cpio:古老的存档工具……144
9.6.3 dd: 处理位流……145
9.7 在一卷磁带上使用多个文件……145
9.8 Bacula……146
9.8.1 Bacula 的模型……146
9.8.2 设置 Bacula……147
9.8.3 安装数据库和 Bacula 的守护进程……148
9.8.4 配置 Bacula 的守护进程……148
9.8.5 bacular-dir.conf: 配置控制文件……148
9.8.6 bacula-sd.conf: 配置存储守护进程……152
9.8.7 bconsole.conf: 配置控制台……152
9.8.8 安装和配置客户端的文件守护进程……153
9.8.9 启动 Bacula 的守护进程……154
9.8.10 向存储池添加介质……154
9.8.11 执行一次手工备份……154
9.8.12 执行一次恢复工作……156
9.8.13 监视和调试 Bacula 的配置……158
9.8.14 Bacula 的替换工具……160
9.9 商用备份产品……160
9.9.1 ADSM/TSM……160
9.9.2 Veritas……161
9.9.3 其他选择……161
9.10 推荐读物……161
9.11 习题……161
第 10 章 系统日志与日志文件……163
10.1 日志记录的策略……163
10.1.1 扔掉日志文件……163
10.1.2 轮换日志文件……164
10.1.3 存档日志文件……165
10.2 Linux 的日志文件……165
10.2.1 特殊的日志文件……167
10.2.2 内核和启动日志……167
10.3 logrotate: 管理日志文件……168

www.TopSage.com

10.4 syslog：系统事件的日志程序……170
10.4.1 syslog 的其他替代方案……170
10.4.2 syslog 的体系结构……170
10.4.3 配置 syslogd……171
10.4.4 设计站点日志方案……173
10.4.5 配置文件举例……174
10.4.6 syslog 输出的样本……175
10.4.7 使用 syslog 的软件……176
10.4.8 调试 syslog……177
10.4.9 在程序中使用 syslog……177
10.5 分析日志文件……178
10.6 习题……179
第 11 章 软件和配置管理……181
11.1 Linux 基本安装……181
11.1.1 从网络引导 PC……182
11.1.2 为 Linux 设置 PXE……183
11.1.3 非 PC 的网络引导……183
11.1.4 Kickstart：RHEL 和 Fedora 的自动安装程序……184
11.1.5 AutoYaST：SUSE 的自动安装工具……186
11.1.6 Debian 和 Ubuntu 的安装程序……186
11.1.7 从主控系统安装……187
11.2 无盘客户机……188
11.3 软件包管理……189
11.3.1 软件包管理系统……189
11.3.2 **rpm**：管理 RPM 软件包……190
11.3.3 **dpkg**：管理 Debian 的软件包……191
11.4 高级软件包管理系统……191
11.4.1 软件包的库……192
11.4.2 RHN：Red Hat 网络……193
11.4.3 APT：高级软件包工具……194
11.4.4 配置 **apt-get**……195
11.4.5 /etc/apt/sources.list 文件的例子……195
11.4.6 使用代理扩展 apt-get……196
11.4.7 设置内部 APT 服务器……196
11.4.8 自动执行 apt-get……197
11.4.9 **yum**：管理 RPM 的发布……198
11.5 版本控制……198
11.5.1 创建备份文件……199
11.5.2 正式的版本控制系统……199
11.5.3 RCS：版本控制系统……200
11.5.4 CVS：并发版本系统……202
11.5.5 Subversion：做得好的 CVS……203
11.6 本地化和配置……205

www.TopSage.com

11.6.1 本地化的组织 ····206
11.6.2 测试 ····206
11.6.3 本地编译软件 ····207
11.6.4 发布本地软件 ····208
11.6.5 解决时间安排上的问题 ····208
11.7 配置管理工具 ····208
11.7.1 cfengine: 计算机免疫系统 ····209
11.7.2 LCFG: 大规模配置系统 ····209
11.7.3 Arusha 项目（ARK） ····209
11.7.4 Template Tree 2: cfengine 的帮手 ····210
11.7.5 DMTF/CIM: 公共信息模型 ····210
11.8 通过 NFS 共享软件 ····211
11.8.1 软件包的名字空间 ····211
11.8.2 依赖关系的管理 ····212
11.8.3 封装脚本 ····212
11.8.4 实现工具 ····212
11.9 推荐软件 ····213
11.10 推荐读物 ····214
11.11 习题 ····215

第二部分 网络管理技术 ····216

第 12 章 TCP/IP 网络 ····217
12.1 TCP/IP 和 Internet ····218
12.1.1 简要的历史回顾 ····218
12.1.2 当今 Internet 的管理方法 ····219
12.1.3 网络标准和文献 ····219
12.2 连网技术概述 ····220
12.3 分组和封装 ····221
12.3.1 链路层 ····222
12.3.2 包的寻址方式 ····223
12.3.3 端口 ····224
12.3.4 地址类型 ····224
12.4 IP 地址详解 ····225
12.4.1 IP 地址分类 ····225
12.4.2 子网和网络掩码 ····225
12.4.3 IP 地址危机 ····227
12.4.4 CIDR: 无类域间路由 ····228
12.4.5 地址分配 ····230
12.4.6 私用地址和 NAT ····230
12.4.7 IPv6 寻址方式 ····232
12.5 路由选择 ····233
12.5.1 路由表 ····233
12.5.2 ICMP 重定向 ····234

www.TopSage.com

12.6 ARP：地址解析协议……235
12.7 将一台主机加入网络……236
12.7.1 分配主机名和 IP 地址……237
12.7.2 ifconfig：配置网络接口……238
12.7.3 mii-tool：配置自动协商和其他介质特有的选项……240
12.7.4 route：配置静态路由……240
12.7.5 默认路由……242
12.7.6 配置 DNS……242
12.7.7 Linux 连网协议栈……243
12.8 特定于发行版本的网络配置……243
12.8.1 Red Hat 和 Fedora 的网络配置……244
12.8.2 SUSE 的网络配置……245
12.8.3 Debian 和 Ubuntu 的网络配置……246
12.9 DHCP：动态主机配置协议……246
12.9.1 DHCP 软件……247
12.9.2 DHCP 的工作方式……247
12.9.3 ISC 的 DHCP 服务器……248
12.10 Linux 的动态重新配置和调整……249
12.11 安全问题……250
12.11.1 IP 转发……251
12.11.2 ICMP 重定向……251
12.11.3 源路由……251
12.11.4 广播 ping 和其他形式的定向广播……251
12.11.5 IP 欺骗……251
12.11.6 基于主机的防火墙……252
12.11.7 虚拟私用网络……252
12.11.8 与安全有关的内核变量……253
12.12 Linux 的 NAT……253
12.13 PPP：点对点协议……254
12.13.1 解决 PPP 的性能问题……254
12.13.2 用 PPP 连网……254
12.13.3 让主机使用 PPP……254
12.13.4 控制 PPP 链路……255
12.13.5 分配地址……255
12.13.6 路由选择……255
12.13.7 确保安全性……255
12.13.8 使用对话脚本……256
12.13.9 Linux 上的 PPP 配置……256
12.14 Linux 连网配置的特殊之处……261
12.15 推荐读物……262
12.16 习题……263
第 13 章 路由选择……265
13.1 近观包转发……266

www.TopSage.com

13.2 路由守护进程和路由协议……268
13.2.1 距离向量协议……268
13.2.2 链路状态协议……269
13.2.3 代价度量……269
13.2.4 内部和外部协议……270
13.3 内部路由协议巡礼……270
13.3.1 RIP：路由信息协议……270
13.3.2 RIP-2：路由选择信息协议第 2 版……271
13.3.3 OSPF：开放最短路径优先……271
13.3.4 IGRP 和 EIGRP：内部网关路由协议……271
13.3.5 IS-IS：ISO 的“标准”……271
13.3.6 MOSPF、DVMRP 和 PIM：多播路由协议……272
13.3.7 路由器发现协议……272
13.4 routed：使用 RIP 的路由守护进程……272
13.5 gated：更好的路由守护进程……273
13.6 路由策略的选择标准……273
13.7 Cisco 路由器……274
13.8 推荐读物……276
13.9 习题……276
第 14 章 网络硬件……278
14.1 选择 LAN、WAN、还是 MAN……278
14.2 以太网：常见的局域网……279
14.2.1 以太网的工作方式……280
14.2.2 以太网拓扑结构……280
14.2.3 无屏蔽双绞线……281
14.2.4 连接和扩展以太网……282
14.3 无线：流动人士的 LAN……284
14.3.1 无线网络的安全……285
14.3.2 无线交换机……286
14.4 FDDI：令人失望且昂贵的局域网……286
14.5 ATM：有前途的（但被痛苦地击败的）LAN……287
14.6 帧中继：被牺牲掉的 WAN……288
14.7 ISDN：本土上的 WAN……288
14.8 DSL 和 CM：大众的 WAN……289
14.9 网络的前景……289
14.10 网络测试和调试……290
14.11 建筑物布线……290
14.11.1 UTP 电缆的选择……290
14.11.2 到办公室的连接……291
14.11.3 布线标准……291
14.12 网络设计问题……292
14.12.1 网络结构与建筑物结构……292
14.12.2 现成的网络……292

www.TopSage.com

14.12.3 扩展……292
14.12.4 拥塞……293
14.12.5 维护和建档……293
14.13 管理问题……293
14.14 推荐的厂商……294
14.14.1 电缆和连接器……294
14.14.2 测试仪器……294
14.14.3 路由器/交换机……294
14.15 推荐读物……295
14.16 习题……295
第15章 域名系统……296
15.1 DNS速成：添加新机器……297
15.2 DNS的历史……298
15.2.1 BIND的实现……298
15.2.2 其他的DNS实现……298
15.3 谁需要DNS……299
15.4 DNS名字空间……300
15.4.1 域名管理……302
15.4.2 选择域名……303
15.4.3 域名爆炸……303
15.4.4 注册二级域名……303
15.4.5 创建子域……304
15.5 DNS的工作原理……304
15.5.1 授权……304
15.5.2 缓存和效率……305
15.5.3 扩展的DNS协议……306
15.6 DNS的新特性……306
15.7 DNS数据库……308
15.7.1 资源记录……308
15.7.2 SOA记录……310
15.7.3 NS记录……312
15.7.4 A记录……313
15.7.5 PTR记录……313
15.7.6 MX记录……314
15.7.7 CNAME记录……316
15.7.8 巧用CNAME……316
15.7.9 LOC记录……317
15.7.10 SRV记录……318
15.7.11 TXT记录……319
15.7.12 IPv6资源记录……320
15.7.13 IPv6正向记录……320
15.7.14 IPv6反向记录……320
15.7.15 与安全有关的记录……321

www.TopSage.com

15.7.16 区文件中的命令……321
15.7.17 粘合记录: 区之间的链接……322
15.8 BIND 软件……324
15.8.1 BIND 的版本……324
15.8.2 确定现有的 BIND 版本……324
15.8.3 BIND 的组成……326
15.8.4 named: BIND 的名字服务器……326
15.8.5 权威与仅缓存服务器……326
15.8.6 递归和非递归服务器……327
15.8.7 解析器库例程……328
15.8.8 DNS 的 shell 接口……328
15.9 设计 DNS 环境……328
15.9.1 名字空间管理……329
15.9.2 权威服务器……329
15.9.3 缓存服务器……330
15.9.4 安全……330
15.9.5 总结……330
15.9.6 DNS/BIND 管理工作分析……331
15.10 BIND 客户机问题……331
15.10.1 配置解析器……331
15.10.2 测试解析器……332
15.10.3 对系统其余部分的影响……332
15.11 配置 BIND 服务器……333
15.11.1 硬件要求……333
15.11.2 配置文件……333
15.11.3 include 语句……335
15.11.4 options 语句……335
15.11.5 acl 语句……340
15.11.6 key 语句……340
15.11.7 trusted-keys 语句……340
15.11.8 server 语句……341
15.11.9 masters 语句……342
15.11.10 logging 语句……342
15.11.11 zone 语句……342
15.11.12 controls 语句……345
15.11.13 分离式 DNS 和 view 语句……346
15.12 BIND 配置举例……347
15.12.1 localhost 区……348
15.12.2 一家小型的安全公司……349
15.12.3 ISC……352
15.13 启动 named……353
15.14 更新区文件……354
15.14.1 区传送……354

www.TopSage.com

15.14.2 动态更新……355
15.15 安全问题……357
15.15.1 再谈访问控制列表……357
15.15.2 限制 named……359
15.15.3 使用 TSIG 和 TKEY 保障服务器与服务器之间通信的安全……359
15.15.4 DNSSEC……361
15.15.5 否定回答……367
15.15.6 微软和 DNS……367
15.16 测试和调试……369
15.16.1 日志功能……369
15.16.2 日志配置样例……372
15.16.3 调试级别……373
15.16.4 用 rndc 调试……373
15.16.5 BIND 统计……374
15.16.6 用 dig 调试……375
15.16.7 残缺授权……376
15.16.8 doc: 域名模糊控制……377
15.16.9 其他 DNS 检查工具……378
15.16.10 性能问题……379
15.17 发行版本的特定信息……379
15.18 推荐读物……381
15.18.1 邮件列表和新闻组……381
15.18.2 书籍和其他文档……381
15.18.3 网上资源……382
15.18.4 RFC……382
15.19 习题……382
第 16 章 网络文件系统……383
16.1 NFS 概述……383
16.1.1 NFS 协议的版本……383
16.1.2 传输协议的选择……384
16.1.3 文件上锁机制……384
16.1.4 磁盘配额……385
16.1.5 cookies 与无状态安装……385
16.1.6 共享文件系统的命名规则……385
16.1.7 安全性与 NFS……385
16.1.8 root 访问与 nobody 账号……386
16.2 服务器端 NFS……387
16.2.1 exports 文件……387
16.2.2 nfsd: 提供文件服务……389
16.3 客户端 NFS……389
16.3.1 在启动时安装远程文件系统……391
16.3.2 端口安全限制……392
16.4 nfsstat: 转储 NFS 统计信息……392

www.TopSage.com

16.5 专用 NFS 文件服务器……392
16.6 自动安装……393
16.6.1 automount：根据需要安装文件系统……394
16.6.2 主控文件……394
16.6.3 映射文件……395
16.6.4 可执行的映射文件……395
16.7 推荐读物……396
16.8 习题……396
第 17 章 共享系统文件……397
17.1 共享什么……398
17.2 nscd：缓存查找的结果……399
17.3 把文件复制到各处……399
17.3.1 rdist：推文件……400
17.3.2 rsync：更安全地传输文件……401
17.3.3 拉文件……403
17.4 NIS：网络信息服务……404
17.4.1 NIS 的工作方式……405
17.4.2 NIS 的优缺点……406
17.4.3 制订管理信息源的优先级……407
17.4.4 网络组……408
17.4.5 建立 NIS 域……409
17.5 LDAP：轻量级目录访问协议……411
17.5.1 LDAP 数据的结构……411
17.5.2 LDAP 的特点……412
17.5.3 LDAP 的文档和规范……413
17.5.4 OpenLDAP：Linux 的 LDAP……413
17.5.5 用 LDAP 代替 NIS……414
17.5.6 LDAP 和安全……415
17.6 推荐读物……415
17.7 习题……415
第 18 章 电子邮件……416
18.1 邮件系统……418
18.1.1 用户代理……419
18.1.2 传输代理……419
18.1.3 投递代理……420
18.1.4 消息库……420
18.1.5 访问代理……420
18.1.6 邮件提交代理……420
18.2 剖析邮件消息……421
18.2.1 邮件寻址……421
18.2.2 阅读邮件信头……422
18.3 邮件基本原理……425
18.3.1 使用邮件服务器……426

www.TopSage.com

18.3.2 使用邮件之家……427
18.3.3 使用 IMAP 或 POP……427
18.4 邮件别名……428
18.4.1 从文件中获取邮递列表……430
18.4.2 发邮件给文件……430
18.4.3 发邮件给程序……431
18.4.4 别名举例……431
18.4.5 邮件转发……432
18.4.6 散列的别名数据库……433
18.5 邮递列表和实现清单的软件……434
18.5.1 维护邮递列表的软件包……434
18.5.2 LDAP：轻量级目录访问协议……436
18.6 sendmail：电子邮件的指挥……438
18.6.1 sendmail 的版本……439
18.6.2 从 sendmail.org 安装 sendmail……439
18.6.3 在 Debian 和 Ubuntu 系统上安装 sendmail……442
18.6.4 开关文件……442
18.6.5 运行模式……442
18.6.6 邮件队列……443
18.7 sendmail 的配置……445
18.7.1 使用 m4 预处理器……445
18.7.2 sendmail 的配置片段……446
18.7.3 从.mc 样板文件构建配置文件……447
18.7.4 改变 sendmail 的配置……448
18.8 sendmail 基本配置原语……448
18.8.1 VERSIONID 宏……448
18.8.2 OSTYPE 宏……449
18.8.3 DOMAIN 宏……450
18.8.4 MAILER 宏……451
18.9 sendmail 高级配置原语……451
18.9.1 FEATURE 宏……452
18.9.2 use_cw_file 功能……452
18.9.3 redirect 功能……452
18.9.4 always_add_domain 功能……453
18.9.5 nocanonify 功能……453
18.9.6 表和数据库……454
18.9.7 mailertable 功能……455
18.9.8 genericstable 功能……455
18.9.9 virtusertable 功能……456
18.9.10 ldap_routing 功能……456
18.9.11 伪装和 MASQUERADE_AS 宏……457
18.9.12 MAIL_HUB 和 SMART_HOST 宏……459
18.9.13 伪装和路由……459

www.TopSage.com

18.9.14 nullclient 功能……460
18.9.15 local_lmtp 和 smrsh 功能……460
18.9.16 local_procmail 功能……461
18.9.17 LOCAL_*宏……461
18.9.18 配置选项……461
18.10 sendmail 中与垃圾邮件相关的功能……463
18.10.1 中继转发……464
18.10.2 访问数据库……465
18.10.3 把用户或站点列入黑名单……467
18.10.4 信头检查……468
18.10.5 速率和连接限制……469
18.10.6 Slamming……470
18.10.7 邮件过滤……470
18.10.8 处理垃圾邮件……471
18.10.9 SpamAssassin……471
18.10.10 SPF 和 Sender ID……471
18.11 配置文件举例……472
18.11.1 sendmail.com 的邮件客户机……472
18.11.2 sendmail.com 的邮件主机……473
18.12 安全与 sendmail……475
18.12.1 所有权……475
18.12.2 权限……476
18.12.3 发到文件和程序的可靠邮件……477
18.12.4 隐私选项……477
18.12.5 运行 chroot 后的 sendmail……478
18.12.6 拒绝服务攻击……479
18.12.7 伪造……479
18.12.8 消息的隐私……480
18.12.9 SASL：简单的身份验证和安全层……480
18.13 sendmail 的性能……481
18.13.1 投递方式……481
18.13.2 队列分组和信封分割……481
18.13.3 队列运行器……482
18.13.4 负载均衡控制……483
18.13.5 队列中无法投递的消息……483
18.13.6 内核调优……484
18.14 sendmail 的统计信息、测试和调试……484
18.14.1 测试和调试……485
18.14.2 啰嗦的投递……486
18.14.3 用 SMTP 通信……487
18.14.4 队列监视……487
18.14.5 日志机制……488
18.15 Exim 邮件系统……489

www.TopSage.com

18.15.1 历史……489
18.15.2 Linux 上的 Exim……489
18.15.3 配置 Exim……490
18.15.4 Exim 和 sendmail 的相似之处……490
18.16 Postfix 邮件系统……491
18.16.1 Postfix 体系结构……491
18.16.2 接收邮件……492
18.16.3 邮件队列管理器……492
18.16.4 发邮件……492
18.16.5 安全……492
18.16.6 Postfix 命令和文档……493
18.16.7 配置 Postfix……493
18.16.8 main.cf 的内容……493
18.16.9 基本设置……494
18.16.10 使用 postconf……494
18.16.11 查找表……495
18.16.12 本地投递……496
18.16.13 虚拟域……496
18.16.14 虚拟别名域……497
18.16.15 虚拟邮箱域……497
18.16.16 访问控制……498
18.16.17 访问表……499
18.16.18 客户机身份验证……500
18.16.19 反垃圾邮件和病毒……500
18.16.20 黑洞列表……501
18.16.21 SpamAssassin 和 promail……501
18.16.22 策略守护进程……502
18.16.23 内容过滤……502
18.16.24 调试……502
18.16.25 查看邮件队列……503
18.16.26 soft_bounce……503
18.16.27 测试访问控制……503
18.17 推荐读物……504
18.18 习题……505
第 19 章 网络管理和调试……507
19.1 网络故障的检测……508
19.2 ping：检查主机是否正常……508
19.3 traceroute：跟踪 IP 包……510
19.4 netstat：获得网络统计信息……511
19.4.1 检查接口的配置信息……512
19.4.2 监视网络连接的状态……513
19.4.3 标识正在监听的网络服务……513
19.4.4 检查路由表……514

www.TopSage.com

19.4.5 查看各种网络协议运行的统计信息……514
19.5 SAR：检查工作接口的活动……515
19.6 包探测器……516
19.6.1 tcpdump：探测器之王……517
19.6.2 Wireshark：可视化的探测器……517
19.7 网络管理协议……518
19.8 SNMP：简单网络管理协议……519
19.8.1 SNMP 的组织结构……519
19.8.2 SNMP 协议操作……520
19.8.3 RMON：远程监视 MIB……520
19.9 NET-SNMP 代理程序……521
19.10 网络管理应用程序……522
19.10.1 NET-SNMP 工具……522
19.10.2 SNMP 数据的采集和绘图……523
19.10.3 Nagios：基于事件的 SNMP 和服务监视工具……524
19.10.4 商业管理平台……524
19.11 推荐读物……525
19.12 习题……525
第 20 章 安全……527
20.1 Linux 安全吗……528
20.2 安全性是如何受损害的……529
20.2.1 社交工程……529
20.2.2 软件漏洞……529
20.2.3 配置错误……530
20.3 认证和标准……530
20.3.1 认证……531
20.3.2 标准……531
20.4 安全的技巧和思想……532
20.4.1 包过滤……532
20.4.2 不必要的服务……533
20.4.3 软件补丁……533
20.4.4 备份……533
20.4.5 口令……533
20.4.6 警惕性……533
20.4.7 普遍原则……533
20.5 /etc/passwd 和/etc/shadow 文件里的安全问题……534
20.5.1 口令的检查和选择……534
20.5.2 口令时限……535
20.5.3 组登录名和共享登录名……535
20.5.4 用户的 shell……535
20.5.5 获得 root 权限的办法……535
20.5.6 PAM：验证奇才……536
20.6 POSIX 能力……537

www.TopSage.com

大家网 TopSage.com

20.7 setuid 程序……537
20.8 重要文件的权限……538
20.9 形形色色的安全事务……538
20.9.1 远程事件日志……538
20.9.2 安全终端……539
20.9.3 /etc/hosts.equiv 和~/.rhosts……539
20.9.4 安全与 NIS……539
20.9.5 安全与 NFS……539
20.9.6 安全与 sendmail……540
20.9.7 安全与备份……540
20.9.8 病毒与蠕虫……540
20.9.9 特洛伊木马……540
20.9.10 Rootkits……541
20.10 加强安全的工具……541
20.10.1 Nmap: 扫描网络端口……541
20.10.2 Nessus: 下一代的网络扫描程序……543
20.10.3 John the Ripper: 找出不安全的口令……543
20.10.4 hosts_acces: 主机访问控制……544
20.10.5 Samhain: 基于主机的入侵检测……544
20.10.6 SELinux……545
20.11 加密的安全工具……546
20.11.1 Kerberos: 用于网络安全的统一方法……547
20.11.2 PGP: 很好的私密性……547
20.11.3 SSH: 安全的 shell……548
20.11.4 一次性口令……549
20.11.5 Stunnel……549
20.12 防火墙……551
20.12.1 包过滤防火墙……551
20.12.2 如何过滤服务……551
20.12.3 服务代理防火墙……552
20.12.4 状态检查防火墙……552
20.12.5 防火墙保险吗……553
20.13 Linux 的防火墙功能: IP Tables……553
20.14 VPN……556
20.14.1 IPsec 隧道……557
20.14.2 VPN 够用吗……557
20.15 强化安全的 Linux 发行版本……557
20.16 如何对付站点攻击……558
20.17 安全信息的来源……559
20.17.1 CERT: 卡内基梅隆大学的注册服务商标……559
20.17.2 Security Focus.com 和 BugTraq 邮递列表……559
20.17.3 Crypto-Gram 时事通信……560
20.17.4 SANS: 系统管理、网络和安全协会……560

www.TopSage.com

20.17.5 特定发行版本的安全资源……560
20.17.6 其他邮递列表和网站……560
20.18 推荐读物……561
20.19 习题……562
第 21 章 Web 主机托管和 Internet 服务器……564
21.1 Web 主机托管的基本知识……564
21.1.1 统一资源定位符……565
21.1.2 HTTP 工作原理……565
21.1.3 CGI 脚本：即时生成内容……566
21.1.4 负载平衡……567
21.2 HTTP 服务程序的安装……567
21.2.1 选择服务器软件……568
21.2.2 安装 Apache……568
21.2.3 配置 Apache……569
21.2.4 运行 Apache……570
21.2.5 分析日志文件……570
21.2.6 高性能主机托管的静态内容优化……570
21.3 虚拟接口……570
21.3.1 使用基于名字的虚拟主机……571
21.3.2 配置虚拟接口……571
21.3.3 告诉 Apache 有关虚拟接口的信息……572
21.4 SSL……573
21.4.1 产生签发证书的请求……573
21.4.2 配置 Apache 使用 SSL……574
21.5 缓存和代理服务程序……575
21.5.1 Squid 缓存和代理服务器……575
21.5.2 设置 Squid……576
21.6 匿名 FTP 服务程序的设置……576
21.7 习题……577
第三部分 其他管理技术……578
第 22 章 X 窗口系统……579
22.1 X 显示管理器……580
22.2 运行一个 X 应用程序……581
22.2.1 环境变量 DISPLAY……581
22.2.2 客户机身份验证……582
22.2.3 用 SSH 转发 X 连接……583
22.3 配置 X 服务器……585
22.3.1 Device 段……586
22.3.2 Monitor 段……586
22.3.3 Screen 段……587
22.3.4 InputDevice 段……587
22.3.5 ServerLayout 段……588

www.TopSage.com

22.4 故障排查和调试……589
22.4.1 X的特殊键盘组合……589
22.4.2 正常X服务器出问题……590
22.5 桌面环境简述……591
22.5.1 KDE……592
22.5.2 GNOME……592
22.5.3 KDE和GNOME谁更好……593
22.6 推荐读物……593
22.7 习题……593
第23章 打印……594
23.1 复杂的打印机……595
23.2 打印机的语言……595
23.2.1 PostScript……596
23.2.2 PCL……596
23.2.3 PDF……596
23.2.4 XHTML……597
23.2.5 PJL……597
23.2.6 打印机驱动程序及其对PDL的处理……597
23.3 CUPS的体系结构……598
23.3.1 打印文档……599
23.3.2 查看和操控打印队列……599
23.3.3 多台打印机……599
23.3.4 打印机实例……600
23.3.5 网络打印……600
23.3.6 CUPS的支持协议：HTTP……600
23.3.7 PPD文件……601
23.3.8 过滤器……602
23.4 CUPS服务器的管理……603
23.4.1 设置网络打印服务器……604
23.4.2 自动配置打印机……604
23.4.3 配置网络打印机……604
23.4.4 打印机的配置举例……605
23.4.5 设置打印机的类……605
23.4.6 关闭服务……606
23.4.7 其他配置工作……606
23.4.8 纸型……606
23.4.9 兼容性命令……607
23.4.10 常用的打印命令……608
23.4.11 CUPS的文档……609
23.5 故障排查的技巧……609
23.5.1 CUPS的日志……609
23.5.2 直接打印的问题……610
23.5.3 网络打印的问题……610

www.TopSage.com

23.5.4 发行版本特有的问题……610
23.6 实际使用打印机的问题……611
23.6.1 打印机的选择……611
23.6.2 GDI 打印机……611
23.6.3 双面打印……611
23.6.4 其他打印机配件……612
23.6.5 串口和并口打印机……612
23.6.6 网络打印机……612
23.7 给打印机的其他建议……612
23.7.1 只在必要时才用标题页……612
23.7.2 提供回收站……613
23.7.3 使用预览程序……613
23.7.4 购买便宜的打印机……613
23.7.5 手头多备墨盒……613
23.7.6 注意单页成本……614
23.7.7 考虑打印机记账……614
23.7.8 保护打印机的安全……615
23.8 KDE 的打印功能……615
23.8.1 kprinter: 打印文档……616
23.8.2 Konqueror 和打印……616
23.9 推荐读物……616
23.10 习题……617
第 24 章 维护与环境……618
24.1 维护基础……618
24.2 维护合同……619
24.2.1 现场维护……619
24.2.2 板卡更换维护……619
24.2.3 保修……619
24.3 处理电路板的知识……620
24.3.1 静电……620
24.3.2 重插板卡……620
24.4 监视器……620
24.5 内存模块……620
24.6 预防性维护……621
24.7 环境……622
24.7.1 温度……622
24.7.2 湿度……622
24.7.3 办公室降温……622
24.7.4 机房降温……622
24.7.5 温度监视……623
24.8 电源……624
24.9 机架……624
24.10 数据中心标准……624

www.TopSage.com

24.11 工具……625
24.12 推荐读物……626
24.13 习题……626
第25章 性能分析……627
25.1 做什么可以提高性能……628
25.2 影响性能的因素……629
25.3 全面检查系统性能……629
25.3.1 CPU使用情况分析……629
25.3.2 Linux的内存管理……631
25.3.3 内存使用情况分析……633
25.3.4 磁盘I/O分析……634
25.3.5 选择磁盘I/O调度器……635
25.3.6 sar: 连续采集和报告统计信息……636
25.3.7 oprofile: 全面的剖析器……636
25.4 求助! 系统为何越来越慢……637
25.5 推荐读物……638
25.6 习题……638
第26章 同Windows协作……640
26.1 从Windows登录到Linux系统……640
26.2 远程桌面访问……641
26.2.1 在Windows计算机上运行X服务器程序……641
26.2.2 VNC: 虚拟网络计算技术……642
26.2.3 Windows RDP: 远程桌面协议……642
26.3 运行Windows和类似Windows的应用……643
26.3.1 双重引导……643
26.3.2 另一种选项: OpenOffice.org……644
26.4 在Windows上用命令行工具……644
26.5 Windows遵守电子邮件和Web标准……644
26.6 通过Samba和CIFS共享文件……645
26.6.1 Samba: UNIX的CIFS服务器……645
26.6.2 Samba的安装……646
26.6.3 文件名编码……647
26.6.4 网络邻居浏览……647
26.6.5 用户身份验证……648
26.6.6 基本的文件共享……649
26.6.7 用户组共享……649
26.6.8 用MS DFS做透明重定向……650
26.6.9 **smbclient**: 简单的CIFS客户端……651
26.6.10 smbfs文件系统……651
26.7 用samba共享打印机……652
26.7.1 从Windows安装打印机驱动程序……653
26.7.2 从命令行安装打印机驱动程序……653
26.8 Samba的调试……654

www.TopSage.com

26.9 推荐读物……656
26.10 习题……656
第 27 章 串行设备……657
27.1 串行标准……657
27.2 替代连接器……660
27.2.1 微型 DIN-8 连接器……660
27.2.2 DB-9 连接器……661
27.2.3 RJ-45 连接器……661
27.2.4 RJ-45 连线的 Yost 标准……662
27.3 硬载波和软载波……664
27.4 硬件流控……664
27.5 电缆长度……665
27.6 串行设备文件……665
27.7 setserial：把串口参数通知给驱动程序……665
27.8 串行设备的软件配置……666
27.9 硬件终端的配置……666
27.9.1 登录过程……667
27.9.2 /etc/inittab 文件……667
27.9.3 终端支持：termcap 和 terminfo 数据库……669
27.10 特殊字符和终端驱动程序……670
27.11 stty：设置终端的选项……670
27.12 tset：自动设置选项……671
27.13 怎样解开僵住的终端……672
27.14 调制解调器……672
27.14.1 调制、错误纠正和数据压缩协议……673
27.14.2 minicom：拨出程序……673
27.14.3 双向调制解调器……674
27.15 调试串行线……674
27.16 其他常用的 I/O 端口……674
27.17 习题……675
第 28 章 驱动程序和内核……677
28.1 内核的适应性……678
28.2 添加设备驱动程序……678
28.2.1 设备文件和设备号……679
28.2.2 创建设备文件……679
28.2.3 sysfs：设备……680
28.2.4 设备命名约定……680
28.3 配置内核的原因和方法……681
28.4 调整 Linux 内核参数……681
28.5 构造 Linux 内核……682
28.5.1 内核没问题就不要改它……683
28.5.2 配置内核选项……683
28.5.3 构建 Linux 内核的二进制文件……684

www.TopSage.com

28.6 添加 Linux 设备驱动程序……685
28.7 可加载内核模块……686
28.8 热插拔……688
28.9 设置内核引导参数……688
28.10 推荐读物……689
28.11 习题……689
第 29 章 守护进程……690
29.1 init：基本守护进程……691
29.2 cron 和 atd：调度命令……691
29.3 inetd 和 xinetd：管理守护进程……692
29.3.1 配置 xinetd……692
29.3.2 配置 inetd……694
29.3.3 services 文件……695
29.3.4 portmap：把 RPC 服务映射到 TCP 和 UDP 端口……696
29.4 内核的守护进程……696
29.5 打印守护进程……697
29.5.1 cupsd：CUPS 的调度器……697
29.5.2 lpd：管理打印……697
29.6 文件服务守护进程……698
29.6.1 rpc.nfsd：提供文件服务……698
29.6.2 rpc.mountd：响应安装请求①……698
29.6.3 amd 和 automount：根据需要安装文件系统……698
29.6.4 rpc.lockd 和 rpc.statd：管理 NFS 锁……698
29.6.5 rpciod：缓存 NFS 数据块……698
29.6.6 rpc.rquotad：远程磁盘配额服务……699
29.6.7 smbd：向 Windows 客户端提供文件和打印服务……699
29.6.8 nmbd：NetBIOS 名字服务器……699
29.7 管理数据库守护进程……699
29.7.1 ypbind：定位 NIS 服务器……699
29.7.2 ypserv：NIS 服务器……699
29.7.3 rpc.ypxfrd：传送 NIS 数据库……699
29.7.4 lwresd：轻量级名字解析器库服务器……699
29.7.5 nscd：名字服务高速缓存守护进程……700
29.8 电子邮件守护进程……700
29.8.1 sendmail：传输电子邮件①……700
29.8.2 smtpd：简单邮件传输协议守护进程①……700
29.8.3 popd：基本的邮箱服务器①……700
29.8.4 imapd：高级的邮箱服务器①……700
29.9 远程登录和命令执行守护进程……700
29.9.1 sshd：安全的远程登录服务器①……701
29.9.2 in.rlogind：过时的远程登录服务器①……701
29.9.3 in.telnetd：另一种远程登录服务器①……701
29.9.4 **in.rshd**：远程命令执行服务器①……701

www.TopSage.com

29.10 引导和配置守护进程……701
29.10.1 dhcpd: 动态地址分配ⓘ……701
29.10.2 in.tftpd: 简单文件传送服务器ⓘ……702
29.10.3 rpc.bootparamd: 高级的无盘支持ⓘ……702
29.10.4 hald: 硬件抽象层（HAL）守护进程……702
29.10.5 udevd: 设备连接串行化通报……702
29.11 其他网络守护进程……702
29.11.1 talkd: 网络聊天服务ⓘ……702
29.11.2 snmpd: 提供远程网络管理服务……702
29.11.3 ftpd: 文件传输服务器ⓘ……703
29.11.4 rsyncd: 在多台主机间保持文件同步ⓘ……703
29.11.5 routed: 维护路由表……703
29.11.6 gated: 维护复杂的路由表……703
29.11.7 named: DNS 服务器……703
29.11.8 syslogd: 处理日志消息……703
29.11.9 in.fingerd: 查找用户ⓘ……704
29.11.10 httpd: World Wide Web 服务器ⓘ……704
29.12 NTPD：时间同步守护进程……704
29.13 习题……705
第 30 章 管理政策与策略……706
30.1 皆大欢喜……706
30.2 IT 职能机构的组成……707
30.3 管理的职能……708
30.3.1 领导……708
30.3.2 聘用、解聘和人事管理……709
30.3.3 分配和跟踪任务……711
30.3.4 管理上级……712
30.3.5 化解矛盾……712
30.4 行政的职能……713
30.4.1 销售……714
30.4.2 采购……714
30.4.3 账务……715
30.4.4 人事……715
30.4.5 市场营销……715
30.4.6 行政事务……716
30.5 开发的职能……716
30.5.1 系统构架的原则……717
30.5.2 管理系统的组成……718
30.5.3 系统管理员的工具箱……718
30.5.4 软件工程的原则……719
30.6 运行的职能……719
30.6.1 瞄准最少停机时间……720
30.6.2 依赖文档……720

www.TopSage.com

30.6.3 改变老硬件的用途或者淘汰老硬件……720
30.7 支持工作……721
30.7.1 可用性……721
30.7.2 服务范围……721
30.7.3 技能要求……723
30.7.4 时间管理……723
30.8 建立文档……724
30.8.1 标准化建档……724
30.8.2 硬件标签……725
30.8.3 用户文档……726
30.9 需求跟踪和故障工单系统……726
30.9.1 故障工单系统共同的功能……726
30.9.2 故障工单系统的用户接受程度……727
30.9.3 故障工单系统……727
30.9.4 派发故障工单……728
30.10 灾难恢复……729
30.10.1 备份和离线的信息……729
30.10.2 处理灾难的人员准备……730
30.10.3 电源和 HVAC……730
30.10.4 网络冗余性……731
30.10.5 安全事件……731
30.10.6 世贸中心的二手故事……732
30.11 书面政策……732
30.11.1 安全政策……734
30.11.2 用户政策协定……734
30.11.3 系统管理员政策协定……736
30.12 法律问题（美国）……736
30.12.1 加密……736
30.12.2 版权……737
30.12.3 隐私……738
30.12.4 EULA……739
30.12.5 政策落实……740
30.12.6 控制=义务……740
30.12.7 软件许可证……740
30.12.8 遵守法规……741
30.13 软件专利……742
30.14 标准……743
30.14.1 LSB: Linux 标准库……743
30.14.2 POSIX……743
30.14.3 ITIL: 信息技术接口库……744
30.14.4 COBIT: 信息及相关技术控制目标……744
30.15 Linux 文化……744
30.16 主流的 Linux……745

www.TopSage.com

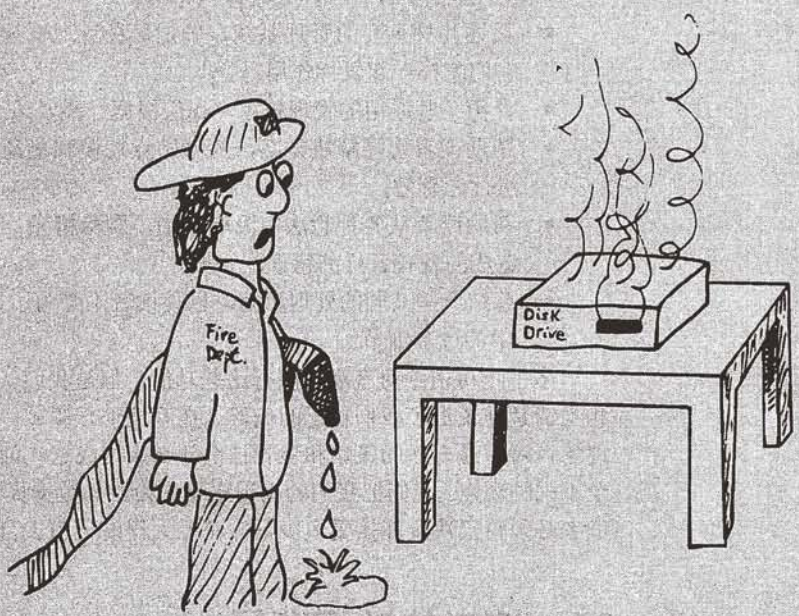
30.17 组织、会议及其他资源……747
30.17.1 大会和商业展览……748
30.17.2 LPI: Linux 专业协会……749
30.17.3 邮递列表和 Web 资源……749
30.17.4 系统管理方面的调查……749
30.18 推荐读物……750
30.18.1 基础设施……750
30.18.2 管理……750
30.18.3 政策和安全……751
30.18.4 法律问题、专利和隐私……751
30.18.5 一般的业界新闻……751
30.19 习题……751

www.TopSage.com

大家网
TopSage.com

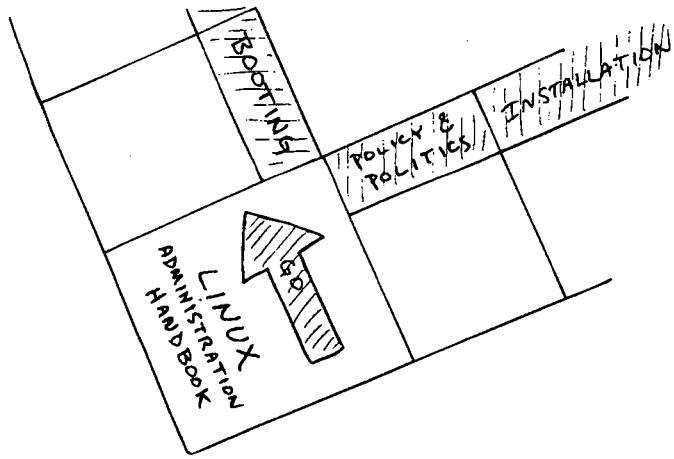
第一部分

基本管理技术



www.TopSage.com

第1章 从何处入手



我们着手编写的是一本可以成为系统管理员所信赖的工作伙伴的书，它不但提供实用建议，而且还包含系统管理的基本理论，读者不可能通过阅读系统手册获得这些知识。因此，本书的目的就是补充而不是替代 Linux 系统文档。

本书以如下 5 种方式向读者提供帮助：

- 概述几种主要的管理系统，确定每种管理系统的不同组成部分，并且解释它们怎样一起发挥作用；
- 介绍一些通用的管理技术，我们根据经验发现这些管理技术既有效又有益；
- 帮助读者选择解决方案，随着站点规模和复杂性的增加，这些解决方案仍然行之有效；
- 帮助读者从各种想法中汲取精华，剔除糟粕，并且告诉读者由版本发布者造成的各式各样的错误；
- 总结一些共同的处理步骤，以使读者不必为了完成简单的任务而深究手册中的过多细节。

本书不可能绝对客观地履行这些功能，但是我们认为，我们已经在字里行间中相当清楚地说出了我们自己的偏好。在系统管理领域，实际中会有这样一个有趣的现象：对于最合适的策略和步骤该由什么来构成这一问题，理性的人们可能会有完全不同的观点。我们把我们的主观意见作为原始数据提供给读者。对于我们的建议，读者必须自己决定能接受多少，并在多大程度上用于自己的环境。

1.1 读者的知识背景

本书假定读者已经具备一定程度的 Linux 或者 UNIX 经验。尤其是在学习管理

www.TopSage.com

知识之前，读者应该从用户的角度对 Linux 的风格有一个一般性的概念，有一些优秀的书籍可以帮助读者迅速掌握这些概念，参见本章 1.10 节。

用户要执行的大多数管理任务都是通过编辑配置文件和编写脚本来完成的，因此读者必须熟悉某种文本编辑器。让许多用户感到沮丧的是，把 Microsoft Word 作为自己惟一的文本编辑器来用，这将会极大地妨碍进行有效的系统管理工作。

我们极力建议读者学会使用 **vi**（在 Linux 系统中最常见的是它的一种经过重写的形式，即 **vim**）。该编辑器是所有 UNIX 和 Linux 系统上的标准软件，尽管同 **emacs** 这样的华丽软件相比，**vi** 可能显得有些平淡，但 **vi** 的功能还是非常强大和完善的。我们也喜欢用 **pico**，这是一种“入门者的编辑器”，简单易学，适合系统管理员新手使用。许多 Linux 发行版本中都包含有 **pico**。选用非标准的编辑器则要小心，如果读者已经对这样的一种编辑器“上了瘾”，那么很快就会对需要把它带在身边以便在每个新系统上安装而感到厌烦。

系统管理工作的主要支柱之一（并且也是贯穿本书的一个主题）是使用脚本来自动完成管理任务。要成为高效的管理人员，必须能够阅读并修改 **Perl** 和 **sh** 脚本（后者在 Linux 下实际上是 **bash** 脚本）。读者从头开始写的脚本可以用 shell 或者自己所选择的脚本语言来编写。

对于编写新脚本，我们推荐使用 **Perl** 或者 **Python**。作为一种编程语言，**Perl** 有些奇怪。不过，它确实包含了许多对于管理员来说是必不可少的特性。本章 1.10 节给出了该书的完整信息。从 cpan.org 可以了解到有关 Perl 语言的所有精选软件。

许多管理员更愿意用 Python 而不是 Perl，而且我们也知道有些站点正在一起努力从 Perl 转向 Python。Python 是一种比 Perl 更精练的语言，而且 Python 脚本往往可读性更好，也更容易维护。

我们还推荐读者学习 **expect**，它与其说是一种编程语言，倒不如说是用于驱动交互式程序执行的前端语言。读者极有可能会很快地学起 **expect** 脚本来。

1.2 Linux 同 UNIX 的关系

在一句话里同时用 Linux 和 UNIX 这两个称谓就仿佛一脚踏入了政治漩涡，或许也像是陷入了一大片泥潭。我们尽可能言简意赅地描述出的现实就是这个样子。

Linux 重新实现了 UNIX，也丰富了 UNIX。Linux 遵循 POSIX 标准，能够在几种硬件平台上运行，兼容现有的大多数 UNIX 软件。它同大多数别的 UNIX 变体的不同之处在于，它是自由的，开放源代码的，而且是由成百上千不同的个人和组织无私奉献协同开发出来的。Linux 融入了在原来的 UNIX 版本中所没有的技术改进，所以它超出了克隆 UNIX 的范畴。从法律意义上讲，它也是一种有别于 UNIX 的东西，严格来说不能称之为“UNIX”。

还值得一提的是，Linux 不是现今世界惟一的自由 UNIX 版本。FreeBSD、NetBSD 和 OpenBSD，以及从 UC Berkeley（加州大学伯克利分校）的 BSD（伯克利软件发布，即 Berkeley Software Distribution）发展而来的所有分支，都有其各自的热诚追随者。虽然这些操作系统从第三方软件厂商得到的支持有点少，但它们在功能和可靠性上却往往可以同 Linux 相媲美。

Linux 软件就是 UNIX 软件。多亏有了 GNU 计划，使 UNIX 系统更富价值的重要软件大多都以某种开放源代码的形式被开发出来。在 Linux 和非 Linux 系统上运行的代码是相同的。比如说，Web 服务器 Apache 全然不在意它是在 Linux 上还是在 HP-UX 上运行的。从应用软件的立场来看，Linux 只不过是得到最好支持的 UNIX 变体之一罢了。

UNIX 和 Linux 系统用于生产环境已经有许多年了。同其他大多数介绍 Linux 系统管理的书不同，

www.TopSage.com

本书着重介绍如何在一个工作环境中——而不只是在单用户桌面环境中——有效地使用 Linux[1]。

1.3 历史上的 Linux

Linux 源自于芬兰研究生 Linus Torvalds 在 1991 年的一项个人计划。他最初构想的这个项目是要作为 Minix 的一个有限分支，Minix 是由 Andrew S. Tanenbaum 编写的一种模型操作系统。但是 Linux 引发了世界上对它的普遍浓厚兴趣，内核很快呈现出了它自己的生命力。利用协作开发的巨大力量，Linus 得以实现更富有雄心的开发进度。1.0 版内核于 1994 年发布，在撰写本书（英文版）的时候（2006 年 9 月），Linux 内核的最新稳定版本是 2.6.17。

因为 Linux 继承了其祖先 UNIX 的太多东西，所以把 Linux 时代的开端算做 1991 年并不算非常公正。UNIX 的历史要回溯到几十年前的 1969 年，它源于当时在 AT&T 贝尔实验室中的一个研究项目。1976 年，UNIX 被无偿向大学发放，因而成了许多操作系统课程和学术研究项目的基础。

1977 年，加州大学伯克利分校的计算机系统研究小组（Computer System Research Group，CSRG）从 AT&T 获得了有许可证的代码，自此成为伯克利 UNIX（Berkeley UNIX）的开端。伯克利的发布版本（称为 BSD，代表 Berkeley Software Distribution）从 1977 年用于 PDP-11 的第一个版本 1BSD 开始，到 1993 年以最高版本 4.4BSD 结束。

随着 UNIX 获得了商业上的认可，源代码许可证的价格迅速攀升。最后，伯克利制订出长期目标，要从 BSD 中去除 AT&T 的代码，这是一个单调乏味而且耗时的过程。在这项工作即将完成时，伯克利失去了研究操作系统的资金支持，于是 CSRG 被解散了。

在解散以前，CSRG 发布了其不含 AT&T 代码的最终软件包，即 4.4BSD-Lite。目前大多数 BSD UNIX 的版本（包括 FreeBSD、NetBSD、Mac OS X[2]和 OpenBSD 等）都宣称 4.4BSD-Lite 软件包是它们的“鼻祖”。

大多数其他的主流 UNIX 版本（包括 HP-UX 和 Solaris）都是有原 AT&T 代码“血统”的后代。Linux 并没有和 AT&T 或者 BSD 风格的 UNIX 共享代码，但是从功能上看，Linux 系统的位置则介于两者之间。

1.4 Linux 的发行版本

Linux 不同于其他 UNIX 变体的原因在于，它只定义了一个操作系统内核。内核必须同命令、守护进程（daemon）以及其他软件一起打包（packaged）来组成一个可以使用的完整操作系统——用 Linux 的术语来说就是一个“发行版本（distribution）”。所有的 Linux 发行版本共享相同的内核源，但是和内核一起的辅助材料则随着发行版本的不同而有很大不同。参考 30.16 节开始的内容了解有关发行版本的更多说明。

上述的“辅助材料”由过去 30 年里成千上万人开发出的大量软件所构成。一直以来都存有这样的争议，即把完整的操作系统简单地称为“Linux”的做法没有让那些开发人员的贡献得到承认，也没有体现出他们工作的历史延续性，这种争议还是有些道理的。遗憾的是，最常被推荐采用的替代称谓“GNU/Linux”也有它自己的历史包袱，所以只有 Debian 这个发行版本正式使用它。在维基百科

[1] 工作环境是一个单位赖以完成实际工作的环境（与之对照的是测试、研究或者开发环境）。

[2] 严格地说，Mac OS X 的内核是 Mach 的一种变体，它是一个混合型的系统，其中既有 BSD 的成分，也有非 UNIX 风格的成分。

www.TopSage.com

“GNU/Linux naming controversy（GNU/Linux 称谓的争议）”这个词条中列出了争执双方各自的理由。

各发行版本在其侧重点、支持度和流行度方面都不尽相同。表 1.1 列举了最流行的通用发行版本。这些发行版本按字母顺序而不是按优先级或者流行度来排列。

表 1.1 最流行的通用 Linux 发行版本

发行版本	Web 站点	说 明
CentOS	www.centos.org	模仿 Red Hat Enterprise Linux 的免费发行版本
Debian	www.debian.org	一种流行的非商业性质的发行版本
Fedora	fedora.redhat.com	从 Red Hat Linux 分出的非商业运作的发行版本
Gentoo	www.gentoo.org	基于源代码的发行版本
Mandriva[a]	www.mandriva.com	对用户最友好的发行版本之一
openSUSE	www.opensuse.org	模仿 SUSE Linux Enterprise 的免费发行版本
Red Hat Enterprise	www.redhat.com	Red Hat Linux 的公司商业化运作的发行版本
Slackware	www.slackware.com	稳定、基础而原始的发行版本
SUSE Linux Enterprise	www.novell.com/Linux	在欧洲流行的多语言发行版本
TurboLinux	www.turbolinux.com	在亚洲流行的多语言发行版本
Ubuntu	www.ubuntu.com	Debian 的精炼版本

a. 以前的 Mandrakelinux。

表 1.1 没有列出许多较小的发行版本，许多没有列出的专用发行版本专门供有特定需要的人们（比如嵌入式系统的开发人员）使用。

Knoppix（www.knoppix.com）是表 1.1 之中没有列出的一种很有用的发行版本，这个版本的 Linux 保存在一张可以启动的 CD-ROM 上。它最重要的价值在于，当某个 Linux 系统因为安全或者技术问题而无法启动的时候，可以用作恢复光盘使用。可启动光盘的概念得到广泛认同，绝大多数发行版本都开始向这个方向转变。现如今的 Ubuntu 就可以从发行光盘启动，Knoppix 的作用变得越发不重要了。从 www.frozentech.com/content/livecd.php 可以找到可启动 Linux 发行版本的最新清单。



在过去将近十年的时间里，Red Hat 公司一直是 Linux 界的领头羊，它的发行版本在北美地区最有影响力。到了 2003 年，Red Hat Linux 发行版本分裂成了两种，一种是以生产应用为中心的产品线，叫做 Red Hat Enterprise Linux（在本书中有时候称之为 RHEL），另一种是依托 Linux 社群的开发项目，叫做 Fedora。在技术、经济、后勤和法律等诸多因素的综合作用下，才促成这次分裂，但时至今日，两方面的发行版本依然保持着相似性。RHEL 提供大量支持，而且稳定性好，但是如果不向 Red Hat 公司支付许可证费，则无法有效地使用它。

CentOS 项目（www.centos.org）收集了 Red Hat 为遵守各种许可证协议（最知名的就是 GNU 的 GPL 许可证）而必须公布的源代码，把这些源代码整理成一个类似于 Red Hat Enterprise Linux，但是却免费的完整发行版本。这个发行版本没有 Red Hat 商标，也没有个别一些专有工具，但是其他各方面都和 RHEL 相似。CentOS 追求同 RHEL 在二进制乃至 bug 上面的完全兼容。

对于那些想要部署一个面向生产应用的发行版本，但又不想向 Red Hat 公司交钱的地方来说，CentOS 是一种很好的选择。采用混合搭配的方式也可行：前端服务器运行 Red Hat Enterprise Linux，使之获得 Red Hat 公司良好的支持，而桌面则运行 CentOS。这样的安排既照顾到让重要的基础系统获得安全保障和技术支持，又将成本和系统管理的复杂度降至最低。

www.TopSage.com



SUSE 现在属于 Novell 公司，最近走上了和 Red Hat 一样的路子，分裂成两种相关的发行版本：一种叫做 openSUSE，只包含自由软件；另一种叫做 SUSE Linux Enterprise，要花钱购买，包含正规的支持渠道，并且多提供了一些额外的东西。以前，SUSE 似乎在有意回避它有免费版本的事实，但是 Novell 已经比其前任更为支持 SUSE 的这个免费版本。现在，读者可以直接访问 www.opensuse.org 获得最新的信息。本书的内容不专门针对哪一种 SUSE 的发行版本，所以我们就直接把它们都统称为“SUSE”。



Debian 和 Ubuntu 这两种发行版本都保持着支持 Linux 社群开发和开放访问的思想意识，所以两者都不存在任何有关自身哪部分自由，哪部分可以重新发布的问题。Debian 在 GNU 社群的热情支持下生存了下来，而 Ubuntu 目前还在享受着南非企业家 Mark Shuttleworth 的慈善资助。Ubuntu 甚至无需读者支付邮费就可以邮寄免费的光盘。

1.4.1 最好的发行版本是什么

在网上随便搜索一下，就会发现这是问得最多、答得最少的 Linux 问题之一。对于读者来说，正确的答案取决于您要怎样使用系统、您所熟悉的 UNIX 变体是什么，以及您对支持的需求。

大多数 Linux 发行版本能够完成您想要让 Linux 系统完成的所有任务。它们中的一些可能要求安装附加的软件来获得完整的功能，有些可能会让某些任务更容易完成。然而，它们之间并不会有太大不同。实际上，为什么会有那么多种不同的发行版本，且每种都宣称“易于安装”和“丰富的软件库”是其突出的特色，这本身就是一件挺奇怪的事情。于是难免会得出这样的结论，人们只不过是喜欢做出新的 Linux 发行版本罢了。

另一方面，既然我们在本书中所关注的是大型 Linux 安装的管理问题，那么我们就会侧重考虑像 Red Hat Enterprise Linux 这样用于管理计算机网络的发行版本。有些发行版本的设计考虑到了生产环境，其他的发行版本则没有。面向生产环境的系统所提供的额外支持功能，对于方便系统管理工作来说，带来了巨大的不同。

当用户采用了一种发行版本之后，就是在某个特定发行商的做事方式上进行了投资。不要只看到软件安装以后的功能，而是要明智地考虑到，自己的单位和那家发行商要在未来的数年里共事。因此要取得一些重要问题的答案。

- 该发行版本能够在今后 5 年内继续存在吗？
- 该发行版本会有最新的安全补丁吗？
- 该发行版本会迅速发布更新软件吗？
- 如果我有问题，发行商会帮助我吗？

有了这样的认识再来看，有些更有意思、规模更小的发行版本就不会太有吸引力了。另一方面，最有生命力的发行版本并不一定是最商业化的发行版本。例如，我们认为 Debian（对，就是 Debian GNU/Linux！）在相当长的一段时间里是会存在的，尽管 Debian 并不是一家公司，既不销售任何东西，也不正式提供需要的支持。

在 www.linux.org/dist 和 distrowatch.com 上都可以找到 Linux 发行版本的完整清单，其中也包括非英语的发行版本。

在本书中，我们采用 5 种流行的 Linux 发行版本作为例子：Red Hat Enterprise Linux 4.3 ES、Fedora Core 5、SUSE Linux Enterprise 10.2、Ubuntu 6.06（“Dapper Drake”）以及当前（到 2006 年 9 月为止）还是测试版的 Debian GNU/Linux 3.2（“Etch”）。这些系统是企业级 Linux 市场的代表，它们合起来占据了如今在大型站点中投入使用的 Linux 的大部分。

www.TopSage.com

1.4.2 特定于发行版本的系统管理工具

许多发行版本提供了可视化的工具（比如 Red Hat 的 Network Administration Tool 或者 SUSE 的 YaST2），帮助用户配置或者管理选定的系统功能。这些工具非常有用，特别是对于管理员新手来说更是如此，但是它们倾向于掩盖用户在进行改动时实际发生的细节。在本书中，出于下述原因，我们要介绍可视化工具调用的底层机制，而不是工具本身。

第一，可视化工具趋于专有化，或者至少是趋于为发行版本所特有，它们给配置过程带来了变数，而在各个发行版本之间，这些过程在较低层面上实际上可能是相当一致的。第二，我们相信，对于系统管理员来说，准确了解他们的系统是如何工作的非常重要。当系统发生故障的时候，可视化工具通常对于确定并修正问题来说没有什么帮助。最后，手工配置往往更好：更快、更灵活、更可靠，而且更易于用脚本实现。

1.5 表示法和印刷约定

在本书中，文件名、命令和命令的参数用粗体（黑体）来表示；占位符（例如，不应该按字面直接照搬的命令参数）用斜体来表示。例如，在下面的命令中：

```
cp file directory
```

用户应该用实际的文件名和实际目录来分别替换 *file* 和 *directory*。

配置文件和终端会话的节录用等宽字体来显示[3]。有时候，我们用斜体字来给交互式会话做注释，例如：

```
$ grep Bob /pub/phonelist          /* 查找 Bob 的电话号码 */
Bob Knowles 555-2834
Bob Smith 555-2311
```

在这些特定情况以外，我们试图把特殊字体和格式约定减到最少，只要我们这样做不会影响理解即可。例如，我们经常会提到的一些词条，比如叫做 daemon 的 Linux 组和打印机 anchor-lw 等，根本不采用特殊格式。

一般说来，我们使用的约定与联机用户手册中用来表示命令的语法相同：

- 中括号（“[” 和 “]”）之间的任何内容都是可选的；
- 英文省略号（“...”）后面的任何内容都是可以重复的；
- 大括号（“{” 和 “}”）表示应该选择由竖线（“|”）隔开的各项中的一个。

例如，规则：

```
bork [-x] {on | off } filename ...
```

可以匹配下面任何一条命令：

```
bork on /etc/passwd
bork -x off /etc/passwd /etc/termcap
```

[3] 实际上，并不是真正的等宽字体，但看起来是的。比起我们所尝试过的真正的等宽字体，我们更喜欢这种安排。这就是为什么在有些示例中，列没有正好对齐的原因。

www.TopSage.com

```
bork off /usr/lib/tmac
```

我们将 shell 风格的特殊字符用于模式匹配：

- 星号（*）匹配零个或多个字符；
- 问号（?）匹配一个字符；
- 中波浪线（~）表示当前用户的主目录（home）；
- *~user* 表示 *user* 用户的主目录。

例如，我们会采用速记模式**/etc/rc*.d** 来指代 Debian 的启动脚本目录**/etc/rc0.d**、**/etc/rc1.d** 等。

引号中的文本经常具有确切的技术含义。在这些情况下，我们忽略英语的一般规则，把标点符号放在引号外边，以使读者不会对什么包括而什么不包括在引号中产生混淆。

特定于系统的信息

除非指定出一种特定发行版本，否则本书中的信息通常适用于我们用来举例的所有发行版本。只适用于某种特定发行版本的细节采用发行商的徽标来标记：


Red Hat® Enterprise Linux® 4.3 ES


Fedora™ Core 5


SUSE® Linux Enterprise 10.2


Ubuntu® 6.06 “Dapper Drake”


Debian® GNU/Linux 3.2 “Etch”（2006 年 9 月的测试版）

使用这些徽标都获得了其各自拥有人的许可。

1.6 从哪里获得信息

Linux 的文档分布在一系列资源中，读者会发现其中有些已经安装到了自己的系统中，而有些还留在网上。较大的资源有：

- 手册页（man 页），用命令 **man** 阅读；
- Texinfo 文档，用命令 **info** 阅读；
- HOWTO，就各种不同主题发表的简短说明；
- Guide，就各种不同主题发表的较长论述；
- 特定于发行版本的文档；
- 同特定的软件项目有关的 Web 网页。

man 手册页和 Texinfo 文档构成了传统的“联机”文档（当然，所有的文档都以这样或者那样的形式在线存在）。这些文档一般伴随系统一起安装，针对特定程序的 man 手册页通常随着新软件包一同安装。

man 手册页对单条命令、驱动程序、文件格式或者库例程做简洁的描述。它们不会解释诸如“我该怎样安装一个新设备？”或者“为什么我们的系统这么慢？”这样较为普通的话题。对于这些问题来说，用户必须参考 HOWTO。

www.TopSage.com

Texinfo 文档是在很久以前由 GNU 人士发明的，针对的是设定 man 页面格式的 **nroff** 命令为 AT&T 专有这一现实情况。现在，我们已经有了 GNU 的 **groff** 来为我们完成这项工作，**nroff** 的问题不再重要了。遗憾的是，许多 GNU 软件包坚持用 Texinfo 文件而不是 man 页面来提供自身的文档。Texinfo 除了为文档规定了不必要的标准之外，其自身也表现为一种相当错综复杂的超文本系统。

要摆脱 Texinfo 的麻烦，可以把 **info** 命令的输出通过管道送给 **less** 命令，避开 **info** 命令内建的浏览体系。这样做还有一个好处，就是能让使用者利用 **less** 命令内置的搜索功能。

幸运的是，采用 Texinfo 提供文档的软件包通常会安装指示性的 man 页面，告诉用户使用 **info** 命令来阅读了解那些特殊的软件包。用户可以坚持使用 **man** 命令来查找手册，只有在被告知要采用 **info** 时再这么做，这样的做法很保险。命令 **info info** 将会把用户带入 Texinfo 的黑暗神秘世界。

HOWTO 和 guide 是由 Linux 建档计划（Linux Documentation Project，LDP）维护的。LDP 是所有各类 Linux 有用信息的中心知识库。它也集中力量将 Linux 的相关文档翻译成其他语言。

特别针对系统管理而言，一些免费的在线 LDP guide 有：Lars Wirzenius、Joanna Oja、Stephen Stafford 和 Alex Weeks 编写的 *The Linux System Administrators' Guide*；Mendel Cooper 编写的 *The Advanced Bash-Scripting Guide*；Olaf Kirch 和 Terry Dawson 编写的 *The Linux Network Administrator's Guide，Second Edition*；以及 Steve Frampton 编写的 *Linux System Administration Made Easy*。

遗憾的是，许多 LDP 文档现在并没有得到持续维护。因为 Linux 的时间概念相对于真实的时间来说要短得多，被忽略掉的文档容易很快过时。一定要查看 HOWTO 或者 guide 上的时间戳，并据此估计它的可信度。

Linux 基本软件中最重要的部分里有许多是由中立的第三方来维护的，比如 Internet 软件联盟（Internet Software Consortium）和 Apache 软件基金会（Apache Software Foundation）。这些组织一般为它们所发布的软件包提供充足的文档。发行版本有时包含了软件，但却给文档打了折扣，于是，到软件原本的发源地看看有没有更多的材料常常会有帮助。

有关许多 Linux 软件包设计原理的另一种有用信息源是“RFC（Request for Comments，请求评议）”系列文档，它们描述了在 Internet 上使用的协议和过程。参考第 12 章了解更多的信息。

1.6.1 手册页的组织

Linux 的手册页一般分成 9 节，如表 1.2 所示。

表 1.2 **Linux 手册页的各节及其内容**

节	内 容	节	内 容
1	用户级命令和应用程序	6	游戏和演示
2	系统调用和内核出错代码	7	各种文件和文档
3	库调用	8	系统管理命令
4	设备驱动程序和网络协议	9	少见隐秘的内核规范和接口
5	标准文件格式		

手册页中有些节会做进一步细分。例如，第 3M 节经常包含了有关系统中数学库的手册页。第 6 节和第 9 节一般是没有内容的。很多系统在手册中都有一个称之为 1 的节，用于本地手册页。另外一个惯用的约定是把第“n”节作为特定于软件的子命令（例如 **bash** 的内建命令）。

手册页的 **nroff** 输入通常保存在**/usr/share/man/manX** 目录中，X 是 **1～9** 的数字，或者是字母 **l** 或 **n**。正常情况下，页面会以 **gzip** 压缩，以节省空间（命令 **man** 知道如何当场将它们解压缩）。手册的格式化版本保存在**/var/cache/man/catX** 目录中。命令 **man** 会随需要确定手册页格式；如果 **cat** 目录

www.TopSage.com

可写，那么 **man** 也将在创建格式化页面的同时，把经过格式化的页面存放到 **cat** 目录中去，从而生成一个常读手册页的高速缓存。

命令 **man** 实际上会在若干不同的目录中寻找用户需要的手册页。用户可以使用 **manpath** 命令来确定搜索路径。这个搜索路径（在 Fedora 中）一般为：

```
$ manpath
/usr/kerberos/man:/usr/local/share/man:/usr/share/man/en:/usr/share/man
```

如果有必要，用户可以设置自己的环境变量 MANPATH 来覆盖默认路径。用户也可以在**/etc/man.config**（RHEL 和 Fedora）或者**/etc/manpath.config**（SUSE、Debian 和 Ubuntu）中设置系统级的默认搜索路径。

1.6.2 man：读取手册页

man *title* 命令格式化特定的手册页并通过 **less** 命令（或者在 PAGER 环境变量中指定的任何程序）把手册页发送到用户终端。*title* 通常是一个命令、设备或文件名。手册中的各节大致是按照数字顺序进行搜索的，不过通常首先搜索描述命令的那些节（第 1、8 和 6 节）。

命令格式 **man** *section title* 可让用户从某个特定的节获取手册页。于是，**man tty** 命令可得到 **tty** 命令的手册页，**man 4 tty** 命令可得到针对控制终端驱动程序的手册页。

man -k *keyword* 输出一份手册页清单，在其单行概要中有 *keyword*。例如：

```
$ man -k translate
objcopy (1)          - copy and translate object files
dcgettext (3)        - translate message
tr (1)               - translate or delete characters
snmptranslate (1) - translate SNMP OID values into more useful information
tr (1p)              - translate characters
gettext (1)          - translate message
ngettext (1)         - translate message and choose plural form
...
```

1.6.3 其他的 Linux 信息资源

在世界上有一个巨大的 Linux 爱好者群体。我们不可能说出每一处有用的 Linux 信息资源，或只是主要的资源，但是在表 1.3 中给出了几个很重要的信息资源。

不要羞于访问普通的 UNIX 资源——大多数信息都能直接用于 Linux。在网上可以获得有关系统管理的丰富信息，形式非常多样。例如，用户可以在任何流行的搜索引擎，比如 Google、Yahoo!或者 Ask 中键入系统管理方面的问题。在第 30 章中可以找到一份其他“入门”资源的清单。

表 1.3 Web 上的 Linux 资源

Web 站点	说 明
linux.slashdot.org	新闻巨头 Slashdot 针对 Linux 的技术支持网站
lwn.net	Linux 和开放源代码方面的新闻集散地
www.freshmeat.net	Linux 和 UNIX 软件的海量索引库
www.kernel.org	Linux 内核的官方网站
www.linux.com	Linux 信息交流地（非官方）
www.linux.org	另一个 Linux 信息交流网站（非官方）

www.TopSage.com

续表

Web 站点	说 明
www.linuxhq.com	有关内核的信息和补丁的汇编
www.linuxworld.com	Computerworld 人士办的在线杂志
www.tucows.com	包含 Linux 内容的多平台软件库

许多站点直接迎合了系统管理员的需求。下面是我们特别喜爱的几个站点：

- www.ugu.com——UNIX Guru Universe（UNIX 高手世界），有许多系统管理的资料；
- www.stokely.com——系统管理资源链接的好集合；
- www.tucows.com——Windows 和 Mac 软件，按质量进行了筛选；
- slashdot.org——“计算机迷”的新闻“集散地”；
- www.cpan.org——Perl 脚本和库的中心资源库；
- securityfocus.com——安全信息，巨大而且可以搜索的安全漏洞数据库。

另一个有趣且有用的资源是 Bruce Hamilton 的“Rosetta Stone”网页，地址为 http://bhami.com/rosetta.html。它提供了在许多不同的操作系统上执行各种系统管理任务所使用的命令和工具的指导。

1.7 如何找到和安装软件

大多数 Linux 发行版本把它们的软件分成可以彼此独立安装的软件包。当用户在一台新的计算机上安装 Linux 的时候，一般要选择一定范围的“起步”软件包，把它们复制到新的系统上。

这种体系简化了系统配置许多方面的工作，也是 Linux 优于 UNIX 传统版本的关键点之一。遗憾的是，这种设计也使得写书介绍这些发行版本的任务复杂了，因为总是不能确切地搞清楚哪些软件是某个给定发行版本的“一部分”。如果一个软件包在安装光盘中，但又不是默认安装的一部分，它还算是该发行版本里的软件包吗？只有在运行那个发行版本的每台计算机上都有这个软件包时，它才算是该发行版本里的软件包吗？如果它只在发行版本的超级版“奉送”光盘上才有，那么它算是该发行版本里的软件包吗？

在本书中，我们一般介绍每个所举例子的发行版本的默认安装。当我们说某个特殊软件包没有包含在默认安装里的时候，并不一定意味着该软件包不会出现在读者的系统上，或者读者的发行版本不支持它。下面就讲一下，如果读者有该软件，如何把它找出来，如果读者没有该软件，那么怎样得到它。

首先，使用 shell 的 **which** 命令来找出相关命令是否已经在搜索路径中。例如，下面的命令揭示出 GNU 的 C 编译器已经安装在了本机的**/usr/bin** 目录下：

```
$ which gcc
/usr/bin/gcc
```

如果 **which** 没有找到要找的命令，那么试试 **whereis**，该命令搜索更大范围的系统目录，和 shell 的搜索路径无关。还要注意，有些系统上的 **which** 命令不显示用户没有执行权限的文件。例如：

```
$ which ipppd
/usr/bin/which: no ipppd in (/bin:/usr/bin:/sbin:/usr/sbin)
$ whereis ipppd
ipppd: /usr/sbin/ipppd
$ ls -l /usr/sbin/ipppd
-rwx------    1 root     root       124924 Aug  3  2000 /usr/sbin/ipppd
```

www.TopSage.com

另一种方法是采取作用非凡的 **locate** 命令，它考察预先编译好的一个文件系统索引，以此确定与特定模式相匹配的文件名。它搜索的并不特定于命令或者软件包，而是能够找到任何类型的文件。例如，如果读者不能确定在哪儿能找到头文件 **signal.h**（它是 Linux 信号定义的正式代码），可以试试：

```
$ locate signal.h
/usr/include/asm/signal.h
/usr/include/linux/signal.h
/usr/include/signal.h
/usr/include/sys/signal.h
```

locate 的数据库通常由 **updatedb** 命令在每天晚上重新生成，这条命令由 **cron** 来运行。因此，执行一次 **locate** 的结果不是总能够反映出文件系统新近的变化。

如果用户知道正在找的软件包的名字，那么也可以使用系统上的软件包工具来直接检查是否有该软件包出现。例如，在 Red Hat、Fedora 或者 SUSE 系统上，下面的命令可以检查是否有 Python 脚本语言：

```
$ rpm -q python
python-1.5.2-27
```

参考第 11 章，了解有关在我们举例的系统中软件打包命令的更多信息。

如果感兴趣的软件包似乎还没有安装，那么找它的第一个地方就是所用发行版本的自动软件包管理系统。每一种发行版本都支持某种形式基于 Internet 的升级旧软件包和查找新软件包的系统。最常见的系统有 yum 和 APT，这两种系统将在本书的 11.4 节介绍。

例如，在使用 APT 的 Debian 系统上，就可以用下面的命令获得并安装最新版的 Python：

```
# apt-get install python
```

大多数 Linux 软件是由独立的小组开发，这些小组以源代码的形式发布软件。接着，Linux 发行商收集源代码，按照其特定系统上采用的规范来编译，并将二进制编译结果打包。安装特定于发行版本的二进制软件包通常要比取得并编译原来的源代码更容易。但是，发行商提供的软件有时候要比当前的版本落后一两个版本。

两个发行商使用相同的软件打包系统不一定意味着两个系统的软件包能够互换使用。例如，Red Hat 和 SUSE 都使用 RPM，但是它们的文件系统布局有点儿不一样。如果能得到针对于您的特殊发行版本而设计的软件包，那么就用它，这始终是最好的做法。

如果其他方式没有找到，那么试试在一个下载站点上查找该软件包，或者用该软件包的名字在 Google 上搜一下。

1.8 系统管理员的基本任务

下面各小节简要介绍了一般要求系统管理员履行的若干工作。这些职责未必是由一个人来执行的，在许多地方，这项工作会分配给几个人。不过，确实至少需要一个人懂得全部工作，并确保有人正在做这些工作。

1.8.1 增加、删除以及管理用户账号

系统管理员负责为新用户增设账号，将不再活动的用户账号删除。增删用户的处理过程可以自动

www.TopSage.com

进行，但在增设新用户之前，仍然必须做出某些管理决定（把用户的主目录放在什么位置、在哪些机器上创建账号等）。参考第 6 章了解有关添加新用户的更多知识。

当某个用户不应该再访问系统时，必须禁用该用户的账号。由该账号拥有的所有文件必须备份到磁带上并保存起来，以使系统不会随着时间的增长，积累不想要的信息。

1.8.2 增删硬件

购买了新硬件或把硬件从一台机器移到另外一台机器上时，必须配置系统，使之识别并使用该硬件。硬件支持的任务可能很简单，例如添加打印机，也可能是复杂得多的任务，例如添加磁盘阵列。参考第 7 章、第 28 章和第 23 章了解有关这些专题的更多知识。

1.8.3 执行备份

执行备份可能是系统管理员最重要的工作了，这也是经常被忽视或者不经心去做的工作。备份工作耗时且单调，但又绝对必要。备份可以自动进行并授权给下属去做，但系统管理员仍然要确保备份工作在按计划正确进行（备份产生的介质真的可以用来恢复文件）。参考第 9 章了解有关备份的更多知识。

1.8.4 安装和更新软件

当获得新软件时，必须安装并测试它，并且经常在几种类型的操作系统和硬件下进行安装和测试。一旦软件能够正常工作，就必须通知用户可以用它，以及它的位置。在补丁和安全升级被发布出来的同时，也必须平稳地将它们融入到本地环境里。

本地软件应该安装到合适的位置，让用户能容易地把它同系统软件区分开来。采用这样的方式可以简化升级操作系统的任务，因为本地软件将不会被升级过程所覆盖。参考第 11 章了解有关软件管理的更多知识。

1.8.5 监视系统

大型设备需要时刻进行监视。日常的监视活动包括确保电子邮件和 Web 服务正常工作、查看日志文件来掌握故障的早期迹象、确保本地网络都连接正确，并监视系统资源（例如磁盘空间）的可用性。

1.8.6 故障诊断

Linux 系统和运行它们的硬件系统偶尔会发生故障。充当技工来诊断故障以及在必要时请来专家排除故障等就成了管理员的工作职责。找出问题往往比修复更为困难。

1.8.7 维护本地文档

系统为适合单位的需要而逐渐改造，随着改造的进行该系统便开始不同于文档当初所描述的那个基本系统了。给整个系统特定于本地环境的各方面建立文档是管理员的职责。这项职责包括为新安装的，但又不是伴随操作系统而来的任何软件编写文档，把电缆的走线位置以及电缆连接方式等信息编写成文档，维护所有硬件维护记录，记录备份状态，编写本地处理过程和策略的文档等。参考 30.8 节了解有关文档的建议。

www.TopSage.com

1.8.8 时刻警惕系统安全

系统管理员必须实施某种安全策略并定期检查，以确保系统安全没有受到侵害。在安全要求低的系统上，这项工作任务可能仅仅包括对非授权访问进行粗略的检查。在安全要求高的系统上，这可能包括仔细布置陷阱和审计程序的监视网。有关安全主题的更详细信息，请参见第 20 章。

1.8.9 帮助用户

尽管帮助用户解决各种问题很少包括在系统管理员工作的规定范围之内，但这通常占据了多数系统管理员日常工作的很大一部分。系统管理员往往遭到从“我的程序昨天正常工作了，今天却不干活了！您改变了什么？”到“我把咖啡洒在键盘上了！我应该泼水到键盘上洗掉咖啡么？”等形形色色的问题的狂轰烂炸。

1.9 重压下的系统管理员

系统管理员通常有多种身份。在现实生活中，系统管理员经常从事其他工作，但同时又被请来管理身边的一些计算机。如果您是处于这种情况，那么也许想要考虑一下这种兼职最终可能出现的情况。

您对您的系统懂得越多，那么用户群体对您的依赖也就越多。网络规模总是在增长，您可能不得不花费更多的时间在系统管理上。很快，您就会发现在您所在的机构中，您成了唯一懂得怎样去执行各种重要任务的人。

一旦同事们开始把您看作是本地系统的管理员，那时要把自己从这种角色中解救出来就很困难了。我们就知道有几个人为了逃避这种状况已经改变了自己的工作。由于许多管理任务是无形中的，您还可能发现自己所在的单位希望您既成为一名全职的系统管理员，同时也成为全职的工程师、作家或秘书。

有些不乐意这样做的系统管理员通过态度不好和服务差劲儿来摆脱这类要求，我们不推荐这种方法。这种方法有损自己的形象，并且可能产生更多问题。

相反，我们建议您把花费在系统管理上的时间用文档的形式写出来，您的目标应该是保持其工作处于可以管理的水平，而且应该搜集让您在申请减轻管理任务时用得上的证据。在大多数单位里，您需要游说管理层半年到一年的时间以后，才可能让自己的管理员位置由别人来取代，因此应该提前计划好。

另一方面，您可能发现自己喜欢系统管理员的工作，并希望成为全职的系统管理员，那么找到一份工作将不会有什么问题。请参考第 30 章，以预先了解可能出现的一些“恐怖”问题。

系统管理员人格综合症

作为系统管理员，由于其工作而导致的不幸却常见的临床症状是“系统管理员人格综合症”。这种症状通常开始于系统管理员职业生涯的第三年年初，并且症状能够持续到退休。特征症状如下所述，但绝不仅限于这些。

- 急性寻呼机幻想恐惧症：烦乱地觉得自己的呼机已经关闭（而实际上没有）、觉得自己平和的夜晚以及其他有意义的事情即将突然结束，要不吃饭来干长达 72 小时马拉松式的工作。
- 用户巫毒绘画症：强制性地给自己的用户群中的某些人创建巫毒娃娃（扎针草人）画像，这

www.TopSage.com

些人似乎不知道他们老是缺乏计划的行为并没有给您的世界带来紧急情况。

- 查看磁带强迫症：深夜出现突如其来的强烈欲望，想要安装备份的磁带以查看磁带是否确实可读、磁带的标签是否正确。
- 科学方法强迫症：强烈地希望去粗暴地摇晃那些看来还没有采用科学方法的从事系统管理的同事。

有许多医疗方法可以用来治愈这些不幸的症状。最有效的是培养良好的幽默感和营造一个小而丰富的办公室酒窖。还可以考虑更具思考意味的方法：任何时候，当有人在您旁边说“服务器是不是又死机了？”时，让自己静静地向远处眺望，并同时敲击自己的脚后跟。如果所有别的治疗方法都不管用，那么就休假吧。

1.10 推荐读物

在出版领域中针对系统管理最好的资源（除去本书之外☺）就是 O'Reilly 公司的系列图书。这个系列的图书从 20 年前的 *UNIX in a Nutshell* 开始，到现在已经给每个重要的 UNIX 和 Linux 子系统以及命令都单独出了书。这个系列还包括有关 Internet、Windows 和其他非 UNIX 专题的图书。所有这些书都价廉物美、内容及时而且针对性强。Tim O'Reilly 对开放源代码运动相当有兴趣，为此专门搞了一个 OSCON 大会，而且还为其他一些技术潮流搞了大会。OSCON 每年召开两次，美国一次，欧洲一次。

虽然市场上有许多介绍性的 Linux 书籍，但是尚未发现值得我们毫无保留地给予推荐的书。一般而言，读者最好找到 UNIX 的“经典著作”。您读到的几乎每样东西都能同等地用到 Linux 上。

SIEVER, ELLEN, AARON WEBER, AND STEPHEN FIGGINS. *Linux in a Nutshell (5th Edition)*. Sebastopol, CA: O'Reilly Media, 2006.

LAMB, LINDA, AND ARNOLD ROBBINS. *Learning the vi Editor (6th Edition)*. Sebastopol, CA: O'Reilly and Associates. 1998.

POWERS, SHELLY, JERRY PEEK, TIM O'REILLY, AND MIKE LOUKIDES. *UNIX Power Tools (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.

WALL, LARRY, TOM CHRISTIANSEN, AND JON ORWANT. *Programming Perl (3rd Edition)*. Cambridge, MA: O'Reilly Media, 2000.

CHRISTIANSEN, TOM, AND NATHAN TORKINGTON. *Perl Cookbook (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.

GANCARZ, MIKE. *Linux and the Unix Philosophy*. Boston: Digital Press, 2003.

SALUS, PETER. *The Daemon, the GNU & the Penguin*. Groklaw. 2006.

1.11 习题

E1.1 了解系统调用 **sync**（不是 **sync** 命令）应该使用什么命令？应该怎样阅读保存在 **/usr/local/share/man** 中的 **sync** 本地手册页。

E1.2 在您的站点上，有一个系统级的配置文件来控制 **man** 的行为吗？如果想要在 **/doc/man** 中保存本地的文档，应该在这个文件中加入哪些行？必须在 **/doc/man** 中使用什么样的目录结构来让它成为 man 手册页层次结构中的一部分？

E1.3 **man** 和 **info** 之间的主要区别是什么？它们各自有什么优点？

www.TopSage.com

TopSage.com

★E1.4 Linux 内核的当前发展状态如何？热点问题是什么？关键人物都有谁？项目是怎样进行管理的？

★E1.5 研究几种 Linux 发行版本（参考 1.4 节的清单，以此作为出发点），为下列每种应用推荐一个发行版本。解释做出选择的理由。

a）在家办公的单个用户；

b）大学的计算机科学实验室；

c）企业的 Web 服务器。

★E1.6 假定您发现了在 Fedora Core 5 提供的文档中说明的 Apache **httpd** 的某种功能并不可行。

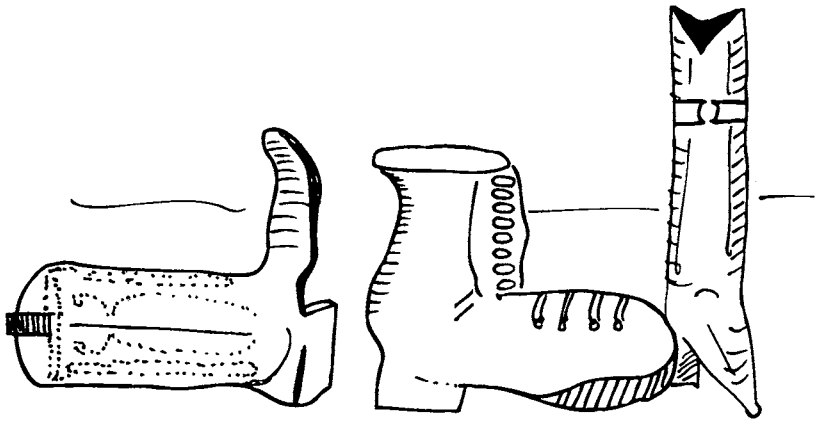
a）在报告这个 bug 之前，您应该做什么？

b）如果认定这确实是一个 bug，那么您应该通知谁？怎样通知？

c）要让 bug 报告有用处的话，它必须包含什么信息？

www.TopSage.com

第 2 章　引导和关机



Linux 是一种复杂的操作系统，比起只是开关电源来说，打开和关闭 Linux 系统要复杂得多。如果要使系统保持正常运行的话，必须正确地执行打开和关闭系统的操作。

尽管系统的引导过程一直带有点儿神秘色彩，但是到了制造商能够控制系统软硬件的所有方面之时，这个过程就比较简单了。既然我们让 Linux 运行在 PC 硬件上，那么引导过程就要按照 PC 的规则来执行，并且要能处理大量不同可能的配置。

虽然这一章出现在本书的前面部分，但是它却援引了书中几百页后才详细讨论的一些内容。具体来说，先熟悉了第 5 章、第 28 章和第 29 章的内容，将会有助于理解本章的知识。如果您的系统已经能够毫无问题地引导起来，那么可以跳过这一章，以后再回过头来学习。

2.1　引导

引导（bootstrapping）是“启动计算机（starting up a computer）”的标准术语。操作系统所提供的正常功能在启动过程中还不能使用，因此，计算机必须“通过其引导程序让自己启动起来”。在引导过程中，内核被加载到内存中并开始执行。各种初始化任务得以执行之后，用户就能够使用系统了。

引导阶段是系统特别脆弱的一段时间。配置文件中的错误、丢失设备或者设备不可靠，以及受损的文件系统都会妨碍计算机的启动。引导配置经常是系统管理员必须在新系统上执行的首批任务之一。遗憾的是，这也是最困难的任务之一，它要求在一定程度上熟悉 Linux 的许多其他方面。

当打开计算机时，计算机执行存储在 ROM 中的引导代码，这些代码接下来尝试确定如何加载并启动内核。内核检测系统的硬件，然后产生系统的 **init** 进程，这个进程总是 PID1（译者注：PID 1 表示进程的 ID 号为 1）。

www.TopSage.com

在出现登录提示符以前，要完成几项工作。系统必须检查并安装文件系统，而且系统的守护进程必须启动起来，这些步骤是由 **init** 进程按顺序运行的一系列 shell 脚本来管理的。启动脚本由于它们的命名方式而经常被称作“rc 文件”，“rc”代表“runcom”或“run command（运行命令）”，这是大约出现于 1965 年的 CTSS 操作系统的历史遗迹。启动脚本的确切位置布局以及它们的执行方式随操作系统的不同而异。我们将在本章后面详细介绍。

2.1.1 自动引导和手工引导

Linux 系统既可以以自动方式也可以以手工方式来引导。在自动方式下，系统自己执行全部引导过程，不需要任何外部的帮助。在手工方式下，系统先自动执行一些过程，然后到某一时刻后，在运行大多数初始化脚本以前，把控制权交给操作员。在这时候，计算机处于“单用户模式”，大多数系统进程还没有运行，其他用户还不能够登录进入系统。

在日常运行中，几乎总是使用自动引导。对于用户来说，在现代机器上典型的引导过程就是打开计算机的电源并等待系统准备就绪。尽管如此，懂得自动引导过程并知道怎样执行手工引导还是很重要的。当出现某些故障而打断了自动引导过程时，例如，出现损坏的文件系统或者出现没有正确配置的网络接口时，通常就不得不采用手工引导。

2.1.2 引导过程的步骤

Linux 系统典型的引导过程由下面 6 个不同的阶段组成：

- 加载并初始化内核；
- 检测和配置设备；
- 创建内核线程；
- 操作员干预（仅用于手工引导）；
- 执行系统启动脚本；
- 多用户模式运行。

系统管理员对以上大多数步骤几乎没有什么控制权。我们通过编辑系统启动脚本来影响大多数的引导配置。

2.1.3 初始化内核

Linux 内核本身就是一个程序，系统引导过程中的第一项任务就是把这个程序载入内存，以便执行它。内核的路径名通常是**/vmlinuz** 或者**/boot/vmlinuz**。参考第 28 章了解有关内核的更多知识。

Linux 系统实现了一种有两个阶段的加载过程。在第一阶段中，系统 ROM 把一个小的引导程序从磁盘载入到内存中。然后，这个程序再安排载入内核。

内核执行内存检测来确定有多少 RAM 可用。内核的一些内部数据结构按静态方式分配其内存量，因此，当内核启动时，它就为自己划分出一块固定大小的实存空间。这块空间保留给内核使用，用户级进程不能使用。内核在控制台上打印一条消息，报告物理内存的总量以及用户进程可用的内存量。

2.1.4 配置硬件

内核执行的第一批任务之一包括检查机器的环境以确定机器有什么硬件。当您为自己的系统构建内核时，要告诉内核，它会找到哪些硬件设备。当内核开始执行时，它试图找到并初始化已经告诉它

www.TopSage.com

的每一个设备。大多数内核为它们所找到的每个设备打印出一行专门信息。现在的发行版本所包含的内核能够在绝大多数机器配置上运行，只要做最少量的定制（如果需要的话）。

在内核配置期间提供的设备信息经常不够明确。在这样的情况下，内核通过探测设备总线和向适当的驱动程序寻求信息来尝试确定它所需要的其他信息。那些没有检测到设备的驱动程序或者那些没有响应探测的驱动程序将被禁用。如果某个设备后来被连接到系统上，那么还是有可能随时加载或者启用它的驱动程序的。参考第 28 章了解细节。

2.1.5 内核线程

一旦完成了基本的初始化任务，内核就在用户空间创建几个“自发”的进程。它们之所以被称作是自发进程，是因为这些进程不是通过系统正规的 **fork** 机制所创建的。详情可参见 4.2 节。

自发进程的数量和特性随系统的不同而不同。在 Linux 上，看不到有 PID 0 进程。和进程 **init**（一定是进程 1）一起的是几个内存和内核处理进程，包括表 2.1 所列出的进程。这些进程的 PID（进程号）都比较小，在 **ps** 命令的输出中它们的名字都被中括号括了起来（例如，[kacpid]）。有时候这些进程的名字以一个斜线加一个数字结尾，比如[kblockd/0]。这个数字表明该线程在哪个处理器上运行，在多处理器的系统上会出现这种有意思的情况。有关 **ps** 的详细介绍参见 4.7 节。

表 2.1 各系统都有一些的 Linux 内核进程

线 程	作 用
kjournald	向磁盘提交 ext3 日志更新信息[a]
kswapd	物理内存不足时执行交换操作的进程
kreclaimd	回收近期未用的内存页
ksoftirqd	处理多层软中断
khubd	配置 USB 设备

a. 每个已经安装的 ext3 文件系统对应一个 kjournald。

在所有这些进程中，只有 **init** 是真正完整的用户进程。其他进程实际上都是内核的组成部分，为了调度或者结构上的原因而进行了装扮，使它们看上去像是进程罢了。

一旦创建完毕自发进程，内核在引导阶段的任务就完成了。不过，处理基本操作（比如接受登录）的进程还一个都没有创建呢，而且大多数 Linux 守护进程也都没有启动。这些任务都是由 **init** 来（有些情况下是间接）负责的。

2.1.6 操作员干预（仅限手工引导）

如果系统以单用户模式进行引导，那么在 **init** 启动时，内核所给出的命令行标志（就是“single”这个单词）会通知 **init** 实际要引导的是单用户模式。进程 **init** 最后会把控制权交给 **sulogin**，后者是 **login** 的一个“中间但不可控”的特殊版本，它提示用户输入 root 口令[1]。如果输入的口令正确，系统将产生一个 root shell。用户可以按下<Control-D>而不是输入口令来绕过单用户模式而继续进入到多用户模式。更详细的信息可参见 2.4 节的内容。参考第 3 章了解更多有关 root 账号的知识。

在单用户 shell 中执行命令的方式和登录到已完全引导的系统上执行命令的方式类似。不过在

[1] 参考 **inittab** 和 **sulogin** 的 man 手册页了解更多信息。糟糕的是，即便是 Red Hat 和 Fedora 当前的版本，在进入单用户模式之前，也不要求输入口令。

www.TopSage.com

SUSE、Debian 还有 Ubuntu 系统上，这时通常只安装了 root 分区。为了使用不在**/bin**、**/sbin** 或**/etc** 下的程序，用户必须手工安装其他文件系统。有关文件系统和安装的更多信息，请参见第 5 章。

在许多单用户环境下，文件系统的根目录是按只读方式安装的。如果/tmp 是根文件系统的一部分，那么许多要使用临时文件的命令（例如 vi）都不能执行。为了解决这个问题，必须先把根文件系统（/）以读写方式重新安装，再开始单用户模式的交互操作。下面这条命令通常就能实现这个技巧。

```
# mount -o rw,remount /
```



Red Hat 和 Fedora 的单用户模式比正常的模式要稍微多做些工作。在出现 *shell* 的命令行提示之前，这两种发行版本都会尝试安装所有的本地文件系统。虽然这样做乍看起来挺有用，但是如果系统中包含一个有错的文件系统，那么就会发生问题。

正常的自动引导过程会运行 **fsck** 命令，检查并修复文件系统。在以单用户模式启动系统时，可能需要手工执行 **fsck**。有关 **fsck** 的更多信息请参见 7.6 节。

当单用户 shell 退出时，系统将尝试继续引导进入多用户模式。

2.1.7 执行启动脚本

到了系统准备运行其启动脚本的时候，我们就能看出它是 Linux 系统了。尽管它看起来还不很像全部启动完毕的系统，但在接下来的启动过程中已经没有什么“奇妙”步骤了。启动脚本就是普通的 shell 脚本，由 **init** 根据一定的算法来选择并运行它们，尽管算法有时候有些复杂，但还是相当容易理解的。

启动脚本的处理、馈送和分类有其自己的专门一节来介绍。本章后面从 2.5 节开始将更详细地介绍这方面的内容。

2.1.8 多用户运行

在初始化脚本运行过以后，系统就是完全运行的系统了，不过现在用户还不能登录进来。为了在某个特定终端（包括控制台）上接受用户登录，必须有一个 **getty** 进程监听终端或者控制台。**init** 直接生成这些 **getty** 进程，完成引导过程。**init** 还负责生成图形登录系统，例如 **xdm** 或 **gdm**（如果设置系统使用它们的话）。有关登录过程的详细信息，请参见 27.9.1 节。

请记住：即使在引导完成以后，**init** 还继续担当重要的角色。**init** 拥有一个单用户和几个多用户“运行级”，运行级决定启用系统的哪些资源。运行级将在本章后面的 2.5 节介绍。

2.2 引导 PC

至此，我们已经看到了引导过程的一般流程。现在我们回顾几个更重要的（并且是更复杂的）步骤。

PC 的引导过程是一次长时间的考验，理解这个过程需要相当多的背景信息。在引导一台机器的时候，它是从执行存储在 ROM 中的代码开始的。根据机器类型的不同，这些代码的确切位置和特性也不相同。在明确为 UNIX 或其他专有操作系统而设计的机器上，这些代码通常是固件，它知道怎样使用连接到机器的设备，知道怎样和网络进行基本通信，还知道怎样理解基于磁盘的文件系统。像这样知道一切的固件对于系统管理员来说是非常方便的。例如，只要键入新内核的文件名，这种固件就知道怎样定位并读取该文件。

www.TopSage.com

在 PC 上，这种初始化的引导代码通常叫做 BIOS（Basic Input/ Output System，基本输入/输出系统），同专有机器的固件相比，它极为简单。实际上，PC 拥有几种级别的 BIOS：一个用于机器本身，一个用于显示卡，另一个用于 SCSI 卡（如果系统有 SCSI 卡的话），有时候还有一个用于其他外设（比如网卡）。

内置的 BIOS 知道主板上一些设备的信息，如 IDE 控制器（和磁盘）、键盘、串口和并口等。SCSI 卡通常只知道与它们相连的设备。幸好让这些设备一起工作所需的复杂交互操作在前几年都已经标准化了，所以几乎无需进行手工干预。

现代的 BIOS 要比以前的稍微聪明一些。它们通常允许用户在启动的时候按下一个或两个特殊键来进入一种配置模式。大多数 BIOS 都可以在启动时告诉用户有哪些特殊键，这样一来用户就不需要再到手册中去查了。

BIOS 通常让用户选择想从什么设备进行引导，这听起来似乎很有用，其实不然。用户通常可以指定“请尝试从软驱引导，然后尝试从 CD-ROM 引导，然后尝试从硬盘引导”诸如此类的顺序。遗憾的是，BIOS 一般被局限于从第一个 IDE CD-ROM 驱动器或第一个 IDE 硬盘引导。以前只有运气非常好，才可能会给您一个能够识别是否有 SCSI 卡的 BIOS。

一旦机器确定从什么设备来启动，那么它将尝试加载磁盘开头 512 个字节的信息。这 512 字节的段叫做 MBR（Master Boot Record，主引导记录）。MBR 包含一个程序，该程序告诉计算机从磁盘的哪个分区加载第二个引导程序（引导加载程序，boot loader）。有关 PC 风格的磁盘分区和 MBR 的更多信息，请参见第 7 章。

默认的 MBR 是一个简单的程序，它告诉计算机从磁盘上的第一个分区获取引导加载程序。Linux 提供了一种更为复杂的 MBR，它知道怎样去处理多操作系统和多内核。

一旦 MBR 已经选定从什么分区进行引导，它就试图加载针对那个分区的引导加载程序。之后，就由引导加载程序负责加载内核。

2.3 引导加载程序：LILO 和 GRUB

没有选择的生活会是什么样子？如今的 Linux 世界里有两种流行的引导加载程序可供选择：LILO 和 GRUB。LILO 是传统的 Linux 引导加载程序，它非常稳定而且有很好的文档，但却迅速被 GRUB 的锋芒盖过。GRUB 已经成为 Red Hat、SUSE 和 Fedora 系统默认的引导加载程序。实际上，如今的 Red Hat 和 Fedora 发行版本根本都不带 LILO。另一方面，Debian 仍然使用 LILO 作为其引导加载程序。

2.3.1 GRUB：全面统一的引导加载程序

对于在同一主机上运行多个操作系统（比如 Windows、OpenBSD、FreeBSD 等）的用户，或者对于积极从事内核开发的用户来说，GRUB（Grand Unified Boot loader）在他们中间格外流行。GRUB 对于频繁改变系统配置的用户来说也很有帮助。每次改变 LILO 的配置之后，必须重新把它安装到引导记录（即 MBR）里，GRUB 与此不同，它会在启动时读入自己的配置文件，省却了上述容易忘记的管理步骤。

用户通过运行 **grub-install** 把 GRUB 安装到引导驱动器上。这条命令的参数是要引导的设备名。GRUB 给物理磁盘命名的方法和 Linux 的习惯标准（虽然 GRUB 也使用标准的 Linux 名字）不一样。遗憾的是，GRUB 自己有一套给物理磁盘设备命名的方法，和标准的 Linux 约定不一样。GRUB 设备名看上去类似于：

www.TopSage.com

```
(hd0,0)
```

第一个数值表示物理驱动器号（从 0 开始），第二个数值表示分区号（也是从 0 开始）。在本例中，(hd0,0)等同于 Linux 设备**/dev/hda1**。因此，如果用户想要在主（primary）设备上安装 GRUB，那么应该使用命令：

```
# grub-install  '(hd0,0)'
```

必须用引号来避免 shell 用它自己的方式来解释括号。

在默认情况下，GRUB 从**/boot/grub/grub.conf** 读取它的默认引导配置。下面是一个 **grub.conf** 文件的例子：

```
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.6.9-5)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.9-5 ro root=/dev/hda1
```

这个例子只配置了一个操作系统，如果 GRUB 在 10s 内（`timeout=10`）没有接收到任何来自键盘的输入，那么就自动引导(`default=0`)。“Red Hat Linux”配置的根文件系统是 GRUB 设备(hd0,0)。GRUB 从**/boot/vmlinuz-2.6.9-5** 加载内核，而且在加载内核的时候显示文件**/boot/grub/splash.xpm.gz** 作为“闪屏”（译者注：即通常所说的启动画面）。

GRUB 支持一种功能强大的命令界面，而且能随时编辑配置文件中的配置项。在 GRUB 的启动画面键入 c 就可以进入命令行模式。从命令行可以启动 **grub.conf** 文件里没有列出的操作系统，可以显示系统信息，还可以执行对文件系统的基础测试。用户还能够在命令行上享受到类似 shell 的特色功能，包括命令补全、光标移动等等。通过 **grub.conf** 文件能做到的事情，也能通过 GRUB 命令行实现。

在命令行上按下<Tab>键就可以看到可能命令的快捷清单。表 2.2 给出了比较有用的一些命令。

表 2.2 GRUB 的命令行选项

命 令	含 义
reboot	软启动系统
find	在所有可以安装的分区上寻找一个文件
root	制定根设备（一个分区）
kernel	从根设备加载的内核
help	获得一条命令的交互性帮助信息
boot	以指定内核影像文件启动系统

要了解有关 GRUB 及其命令行选项的详细信息，可以参考其官方手册：www.gnu.org/software/grub/manual/。

2.3.2 LILO：传统的 Linux 引导加载程序

LILO 要用 **lilo** 命令来配置和安装。**lilo** 命令根据文件**/etc/lilo.conf** 的内容安装配置信息。要改变引导配置，只要更新**/etc/lilo.conf** 并重新运行 **lilo** 即可。引导过程每次发生变化时，都必须重新配置 LILO——具体来说，当用户每次想要添加一个新的引导分区以及每次有一个新的内核要引导时，都

www.TopSage.com

要重新配置 LILO。

下面是 Linux 系统的一个基本的 **lilo.conf** 文件，它既拥有一个工作内核，也有一个备份内核：

```
boot=/dev/hda           # Put boot loader on MBR
root=/dev/hda1          # Specify root partition
install=/boot/boot.b
map=/boot/map
delay=20                # 2 sec for user interrupt
image=/vmlinuz          # Kernel to boot
    label=linux         # Label to refer to this entry
    read-only
image=/vmlinuz-backup # Backup entry
    label=backup
    read-only
```

每种可能的引导情形都有一个标签。在引导时，用户通过输入合适的标签就可以告诉 LILO 使用哪一个标签来引导。出现在 **lilo.conf** 中的第一个标签成为默认的引导标签。

在默认的情况下（名为 `linux`）引导内核文件**/vmlinuz**。“`read-only`”标记指出内核应该以只读方式安装它的根文件系统。一定要有这个选项，启动脚本在适当的时候将以读写方式来重新安装这个分区。这个系统还配置成可以从备份内核**/vmlinuz-backup** 进行引导。有这么一个备选方案总是一个好主意，内核配置受损的话会致使系统无法启动。

不带任何参数去运行 **lilo** 命令将生成并安装引导加载程序，而且告诉用户有哪些引导项可供使用。它在默认的引导映像旁边打一个星号。不过，如果用户已经在 **lilo.conf** 文件中犯了某个错误，**lilo** 通常要到引导加载程序安装到半途中才会发现这个问题。当这种情况发生时，引导加载程序就陷入一种混乱状态。在您成功地运行了 **lilo** 命令之前不要重新启动系统。

为了避免陷入这种状态，可以运行 **lilo -t** 来测试配置而不真的去安装它。如果一切看上去正常，那么就可以运行 **lilo** 进行实际安装。为什么 **lilo** 不默认地运行这个选项来为用户作预先测试呢？这可有点儿奇怪。

针对上述配置文件运行 **lilo**，其输出如下：

```
# lilo
Added linux*
Added backup
```

当系统引导时，LILO 显示下面的提示符：

```
LILO:
```

等待 2 秒（20 个 1/10 秒，用 `delay` 标记来设置），然后引导**/vmlinuz** 内核并把第一个 IDE 硬盘的第一个分区安装为根分区。按下<Tab>键，可以看到已经定义好的引导方案：

```
LILO: <Tab>
linux  backup
LILO:
```

要使用另外一种方案进行引导，只要在提示符处输入它的标签即可。

2.3.3 内核选项

LILO 和 GRUB 都能把命令行选项传给内核。这些选项往往用来修改内核参数的取值，命令内核探测特殊的设备，指定 init 所在的路径，或者指派一个特定的根设备。表 2.3 给出了几个例子。

www.TopSage.com

表 2.3 内核启动时的选项

选 项	含 义
init=/sbin/init	告诉内核用/sbin/init 作为它的 init 程序
init=/bin/bash	只启动 bash，在紧急恢复时有用
root=/dev/foo	告诉内核用/dev/foo 作为根设备
single	引导进入单用户模式

2.3.4 PC 上的多重引导

由于 PC 上可以运行多种操作系统，因此配置一台机器使它能够引导几种不同的系统就成为相当常见的做法。要实现这一点，需要配置一个引导加载程序能认出磁盘上所有不同的操作系统。在接下来的几节中，我们将介绍多重引导常见的一些难点，然后看一些配置的例子。

每个磁盘分区都可以拥有自己的第二阶段的引导加载程序。但是，整个磁盘却只有一个 MBR。在建立多重引导配置时，必须决定哪一个引导加载程序将成为“主”引导加载程序。不管如何，您所做出的选择将取决于涉及到的各个操作系统的特性。对于有一个 Linux 分区的系统来说，LILO 和 GRUB 通常是最佳选择。在多重引导的情况下，GRUB 要比 LILO 更好。

2.3.5 GRUB 的多重引导配置

多重引导的 GRUB 系统和单一引导的 GRUB 系统很相像。先装好所有想要的操作系统，然后再对**/boot/grub/grub.conf** 做相应的修改。

引导 Windows 的 **grub.conf** 配置看上去和引导 UNIX 或者 Linux 系统的配置不一样：

```
title Windows XP
      rootnoverify (hd0,0)
      chainloader +1
```

chainloader 这个选项从一个指定的位置加载引导加载程序（本例中是主 IDE 驱动器上第一个分区的第一个扇区）。`rootnoverify` 选项确保 GRUB 不会尝试安装指定的分区。这个选项避免了 GRUB 被它所不能理解的分区搞糊涂，比如 NTFS 分区或者 GRUB 所能读取的范围之外的其他分区。

下面的 **grub.conf** 文件能够从分区 1 引导 Windows XP，从分区 2 引导 Red Hat Enterprise Linux，还能从分区 3 引导 Fedora：

```
default=0
timeout=5
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
hiddenmenu
title Windows XP
      rootnoverify (hd0,0)
      chainloader +1
title Red Hat
      root (hd0,1)
      kernel /boot/vmlinuz
title Fedora
      root (hd0,2)
      kernel /boot/vmlinuz
```

www.TopSage.com

2.3.6 LILO 的多重引导配置

要配置一个使用 MBR 中 LILO 的多重引导系统（例如，Linux 和 Windows XP 共存的情况），可以拿前面给出的标准 LILO 配置作为基础，然后返回去向**/etc/lilo.conf** 中添加用于其他操作系统的条目。

下面是从第一个 IDE 硬盘的第一个分区引导 Windows 所需的 **lilo.conf** 配置项：

```
other = /dev/hda1
label = windows
table = /dev/hda
```

从第一个分区引导 Windows、从第二个分区引导 Linux、从第三个分区引导 FreeBSD 的完整的 **lilo.conf** 文件如下：

```
boot = /dev/hda          # install on the MBR of 1st IDE drive
delay = 20               # Wait 2 sec. for user's boot choice
default = linux          # If no input, boot linux from 2nd partition
image = /boot/vmlinuz-2.6.9
    root = /dev/hda2
    label = linux
    read-only
other = /dev/hda1        # boot from 1st partition
    label = windows
    table = /dev/hda
other = /dev/hda3        # boot from 3rd partition
    label = freebsd
    table = /dev/hda
```

把这些配置项加到 **lilo.conf** 文件以后，需要重新运行 **lilo** 命令。别忘了先运行 **lilo -t**，测试一下这个配置文件。请参考 7.4 节了解更多有关分区的信息。

发行商（或者是志愿者）经常发布针对 Linux 发行版本的补丁，内核也不例外。安全漏洞、bug 以及新功能都会定期补充到内核中来。但是和其他软件包不一样的是，内核补丁并不对原来内核进行更新，而是与现有的内核平行安装。随着时间的推移，LILO 和 GRUB 的启动菜单中就会充满各种不同版本的内核。使用默认的选择通常是安全的，但是如果系统在打过补丁之后无法启动了，就要意识到可能要简单地对此做些修正。

2.4 引导单用户模式

单用户模式是一种在不影响其他用户（或者不被其他用户影响）的情况下，修改系统配置或者实施维护工作的好方式。单用户模式也是在受损系统上展开恢复工作所采用的模式。

向引导加载程序传递参数是进入单用户模式最常用的做法。不过，用户也可以在其他运行级内通过执行 **telinit 1** 命令进入单用户模式。除非正在调式和引导有关的问题，否则没必要重新启动系统。

文件系统的根目录在系统引导时以只读方式安装，这是一种预防系统不稳定的手段。如果需要通过修改位于根文件系统中的一个配置文件或者命令来解决问题，或者需要执行一条会修改这些文件的命令，那么上述手段就会妨碍我们达到预期的目的。为了解决这个问题，可以用下面的命令重新按读写模式安装根文件系统：

```
# mount -o remount-w /
```

www.TopSage.com

TopSage.com

GRUB 和 LILO 在系统引导过程中进入单用户模式的确切步骤不一样。

2.4.1 用 GRUB 引导单用户模式

在 GRUB 下，不需要使用命令行就能进入单用户模式。GRUB 的作者们认识到，引导选项应该很容易就能修改，并且用“a”键作为相应的工具来设定引导选项。在 GRUB 的启动画面中，加亮显示所需的内核，按下“a”键后添加引导选项即可。要引导单用户模式，就在现有的内核选项之后再加上 single 标志。下面是一个典型的配置例子：

```
grub append> ro root=LABEL=/ rhgb quiet single
```

2.4.2 用 LILO 引导单用户模式

各个发行版本提供了不同的访问 LILO 命令行的途径。如果在偏爱 GRUB 的 Red Hat、Fedora 或者 SUSE 上安装了 LILO，那么就就要从绚丽的全彩色启动界面上选择“command line”这个菜单选项。Debian 和 Ubuntu 的用户则应该在 BIOS 刚执行完内存检查和其他系统自检之后马上按下 shift 键不松手。

在 LILO 提示符处，键入想要引导的配置的标签名（按照 **lilo.conf** 中指定的名字），后跟**-s** 或者 **single**。例如，Debian 自带的默认配置叫做“linux”，所以要引导该配置进入单用户模式，应该使用

```
LILO: linux single
```

2.5 启动脚本

从单用户模式退出以后（或者，在自动引导过程中要运行单用户 shell 的时刻），**init** 执行系统的启动脚本。这些脚本实际上只是由 **sh**（实际是 **bash**）解释的普通 shell 脚本。在不同的系统中，这些脚本的确切位置、内容和组织结构有相当大的区别。

经常在启动脚本中执行的一些任务如下：

- 设置计算机的名称；
- 设置时区；
- 采用 **fsck** 检查磁盘（只在自动模式下）；
- 安装系统的磁盘；
- 从/**tmp** 目录删除旧文件；
- 配置网络接口；
- 启动守护进程和网络服务。

大多数启动脚本的内容相当详尽，而且能打印出正在做的每件事情的描述。如果系统在引导过程中出现问题而挂起，或者用户正试图确定错误在某个脚本中的位置时，这种打印信息的啰嗦做法将会有莫大帮助。

在以前的系统上，系统管理员常做的一件事是修改启动脚本，使得脚本为特定的环境做合适的事情。不过，由于软件打包的粒度更细，而且会频繁从 Internet 进行更新，迫使系统采用了一种更可靠的方式。现在，系统装了由各个软件安装的大量启动小脚本，这些脚本从若干独立的文件读取它们的本地配置信息。这些本地配置文件通常采用微小 **sh** 脚本的形式来设置 shell 变量的值，然后启动脚本再用这些变量的取值。

www.TopSage.com

2.5.1 init 和运行级

传统的 **init** 定义了 7 个“运行级（run level）”，每一个级别都代表系统应该补充运行的某些特定服务：

- 0 级是完全关闭系统的级别；
- 1 级或 S 级代表单用户模式；
- 2～5 级是多用户级别；
- 6 级是“重新引导（reboot）”的级别。

0 级和 6 级比较特殊，因为系统实际上不能留在这两个级别里，进入这两个级别的效果是关闭系统或者重新引导系统。在大多数系统上，正常的多用户运行级别是 2 或 3 级。运行级 5 经常用于 X Windows 的登录进程，比如 **xdm**。运行级 4 很少使用，运行级 1 和 S 的定义在每种系统上是不同的。

单用户模式传统上是 **init** 的级别 1。它关闭所有的多用户和远程登录进程，确保系统运行在最小软件组合的模式下。不过，由于单用户模式提供对系统的超级用户访问权限，因此只要引导系统进入到单用户模式，管理员都要让系统提示用户输入 root 的口令。创建 S 运行级是为了解决下面的需要：它产生一个进程提示输入 root 的口令。在 Linux 上，这一级别只是用来提示输入 root 的口令，而它本身并不是最终的运行目的。

看起来，定出的运行级别比严格需要的或者说有用的级别要多。对这一点的通常解释是：电话交换机（phone switch）有 7 个运行级别，那么想来一个 UNIX 系统至少也应该拥有这么多的运行级吧。Linux 实际上支持多达 10 个运行级，但运行级 7～9 并没有定义。

/etc/inittab 文件告诉 **init** 在它的每个运行级上要做什么事情。它的格式随系统的不同而异，但基本思想是：**inittab** 规定了系统进入到每一级别时要运行（或者要保持运行）的命令。

在机器引导时，**init** 从运行级 0 开始，一级一级往上运行到在**/etc/inittab** 中所设置的默认运行级别。为了完成在每一对相邻运行级别之间的过渡，**init** 运行在**/etc/inittab** 中为这种过渡而说明的一些操作。当机器关闭时，以相反的顺序执行同样的处理过程。

遗憾的是，**inittab** 文件的语义有点儿不那么完善，为了把 **inittab** 文件的功能映射成为某种更为灵活的形式，Linux 系统实现了另一层抽象，它通常采用“改变运行级”脚本的形式（通常为**/etc/init.d/rc**），由 **inittab** 来调用。这一脚本接下来执行位于与运行级有关的目录下的其他脚本，从而把系统带入到新的状态。

现在，大多数 Linux 发行版本默认启动到运行级 5，对于不需要运行 X 的服务器来说这个级别并不合适。默认运行级很容易修改。下面这个从 SUSE 主机上的 **inittab** 文件中截取的片段设置默认启动到运行级 5：

```
id:5:initdefault:
```

系统管理员通常不必直接处理**/etc/inittab**，因为几乎对于任何应用程序来说，基于脚本的接口就足够了。在本章的其余部分里，我们将会按惯例忽略 **inittab** 文件和其他将 **init** 与启动脚本的执行联系起来的文件。只要记住，当我们说 **init** 运行这样那样的脚本时，这种关系可能并不那么直接就行了。

启动脚本的主拷贝位于**/etc/init.d** 这个目录下。每个脚本负责一个守护进程或者系统的某个特定方面。这些脚本都认识参数 **start** 和 **stop**，从而知道它们所处理的服务是应该启动还是应该停止。大多数脚本还认识参数 **restart**，通常该参数等同于在 **stop** 后面再接 **start**。作为系统管理员，要启动和停止各个服务，只要手工运行与之有关的 **init.d** 脚本就可以了。

例如，下面是一个简单的启动脚本，它可以启动、停止或重新启动 **sshd**：

www.TopSage.com

```
#! /bin/sh
test -f /usr/bin/sshd || exit 0
case "$1" in
      start)
            echo -n "Starting sshd: sshd"
            /usr/sbin/sshd
            echo "."
            ;;
      stop)
            echo -n "Stopping sshd: sshd"
            kill `cat /var/run/sshd.pid`
            echo "."
            ;;
       restart)
            echo -n "Stopping sshd: sshd"
            kill `cat /var/run/sshd.pid`
            echo "."
            echo -n "Starting sshd: sshd"
            /usr/sbin/sshd
            echo "."
            ;;
       *)
            echo "Usage: /etc/init.d/sshd start|stop|restart"
            exit 1
            ;;
esac
```

尽管**/etc/init.d** 中的脚本能够启动和停止各个服务，但是由 **init** 运行的主控制脚本需要知道其他一些信息，这些信息说明了要进入任何指定的运行级别需运行哪些脚本（并带什么参数）。当主脚本把系统引入到一个新的运行级别时，它不是直接在 **init.d** 目录下找，而是查找叫做 **rc***level*.**d** 的目录，这里的 *level* 就是要进入的运行级别编号（例如，**rc0.d**、**rc1.d** 等）。

在典型情况下，这些 **rc***level*.**d** 目录包含的符号链接都链接到了 **init.d** 目录中的脚本上。这些符号链接的名称都以 **S** 或 **K** 开头，后跟一个数字以及该脚本所控制的服务名（例如，**S34named**）。当 **init** 从低的运行级别向高的运行级别过渡时，它按照数字递增的顺序运行所有以 **S** 开头的、带有 **start** 参数的脚本。当 **init** 从高的运行级别向低的运行级别过渡时，它按照数字递减的顺序运行所有以 **K**（表示 kill，杀死的意思）开头的、带有 **stop** 参数的脚本。

这一机制让系统管理员可以细粒度地控制启动服务的顺序。例如，在网络接口启动之前先启动 SSH 就没有意义。在 Fedora 系统上，虽然 **network** 和 **sshd** 都配置了在运行级 2 启动，但是 **network** 的脚本序号为 10，而 **sshd** 的脚本序号为 55，所以 **network** 肯定先运行。在加入新服务的时候，一定要考虑到这种依赖关系。要告诉系统什么时候启动一个守护进程，我们必须在适当的目录下创建符号链接。例如，要告诉系统在运行级 2 期间启动 CUPS 并在系统关闭以前妥善地停止这个守护进程，那么创建下面这一对链接就够了：

```
# ln -s /etc/init.d/cups /etc/rc2.d/S80cups
# ln -s /etc/init.d/cups /etc/rc0.d/K80cups
```

第一行命令告诉系统：当进入运行级 2 时，把运行启动脚本**/etc/init.d/cups** 作为最后要做的事情之一，并且带 **start** 参数去运行这个脚本。第二行告诉系统：当关闭系统时，要较早运行**/etc/init.d/cups**，并且带 **stop** 参数来运行这个脚本。有些系统以不同方式处理系统关闭（shutdown）和重新引导（reboot），所以我们需要在**/etc/rc6.d** 目录中也放一个符号链接，以确保当系统重新引导时，该守护进程能够被正确关闭。

www.TopSage.com

2.5.2 Red Hat 和 Fedora 的启动脚本



Red Hat 和 Fedora 的启动脚本历来非常混乱，代码中包含有许多注释，例如：

```
# Yes, this is an ugly, but necessary hack
```

在每个运行级上，**init** 都把新运行级作为参数来调用脚本**/etc/rc.d/rc**。**/etc/rc.d/rc** 一般运行在“正常”模式下，在这种模式下，它只做它自己的事情。它也可以运行在“确认”模式下，在这种模式下，它在运行每个单独的启动脚本以前询问用户。

Red Hat 和 Fedora 有一个 **chkconfig** 命令来帮助用户管理服务。这条命令可以在系统中增删启动脚本，也可以管理这些脚本执行的运行级，还能列出一个脚本目前为哪些运行级做了配置。参考 **man chkconfig** 的输出了解有关这个简便工具的用法。

Red Hat 还有一个 **rc.local** 脚本，和在 BSD 系统上看到的非常类似。**rc.local** 是作为启动过程的一部分而运行的最后一个脚本。以前 **initscripts** 这个软件包会覆盖 **rc.local** 的内容。不过现在已经不这样了，向 **rc.local** 中添加用户自己定制的启动内容也很安全。

下面是 Red Hat 启动会话的一个例子：

```
[kernel information]
INIT: version 2.85 booting
Setting default font (latarcyrhev-sun16):                    [  OK  ]
            Welcome to Red Hat Linux
        Press 'I' to enter interactive startup.
Starting udev:                                               [  OK  ]
Initializing hardware... storage network audio done
Configuring kernel parameters:                               [  OK  ]
Setting clock  (localtime): Tue Mar 29 20:50:41 MST 2005:    [  OK  ]
...
```

一旦看到“Welcome to Red Hat Enterprise Linux”这则消息，用户就可以按“i”键进入“确认”模式。遗憾的是，Red Hat 并没有让用户确认是否已经按下了正确的键。它继续安装本地文件系统、激活交换分区、加载键映射文件并定位它的内核模块。只有当它切换到运行级 3 时，才真正开始提示用户进行确认：

```
      Welcome to Red Hat Enterprise Linux WS
        Press 'I' to enter interactive startup.
Starting udev:                                               [  OK  ]
Initializing hardware... storage network audio done
Configuring kernel parameters:                               [  OK  ]
setting clock  (localtime): tue mar 29 20:50:41 mst 2005:    [  OK  ]
Setting hostname rhel4:                                      [  OK  ]
Checking root filesystem
/dev/hda1: clean, 73355/191616 files, 214536/383032 blocks
                                                             [  OK  ]
Remounting root filesystem in read-write mode:               [  OK  ]
Setting up Logical Volume Management:                        [  OK  ]
Checking filesystems
Mounting local filesystems:                                  [  OK  ]
Enabling local filesystem quotas:                            [  OK  ]
Enabling swap space:                                         [  OK  ]
INIT: Entering runlevel: 3
Entering interactive startup
Start service kudzu (Y)es/(N)o/(C)ontinue? [Y]
```

www.TopSage.com

交互式启动模式和单用户模式在引导过程中的起点是相同的。当启动过程被中断而使得用户不能够安全地到达这个起点时，就可以使用应急软盘或者光盘去引导。

也可以给 LILO 传递参数 **init=/bin/sh**，用技巧让它在 **init** 启动以前运行一个单用户的 shell[2]。如果采用后面这种方案，那么必须手工完成全部的启动工作，包括手工用 **fsck** 命令检查本地文件系统以及安装它们。

Red Hat 引导过程的大多数配置应该通过操作**/etc/sysconfig** 中的配置文件来完成。表 2.4 总结了**/etc/sysconfig** 目录中各项的功能。

表 2.4　　Red Hat 的/etc/sysconfig 目录下的文件和子目录

文件/目录	功能或内容
clock	指定系统有的时钟类型（几乎总是 UTC）[a]
console	一个总是为空的神秘目录
httpd	决定使用 Apache 的何种处理模式
hwconf	包含系统硬件的所有信息，由 Kudzu 使用
i18n	包含系统的本地设置（日期格式、语言等）
init	配置来自启动脚本的消息的显示方式
keyboard	设置键盘类型（使用 us 代表标准的 101 键美国键盘）
mouse	设置鼠标类型，由 X 和 **gpm** 使用
network	设置全局的网络参数（主机名称、网关、转发机制等）
network-scripts	包含补充脚本和网络配置文件的目录
sendmail	为 **sendmail** 设置选项

a. 如果用户的 PC 是多重引导的，那么时钟的时区应该如何设置就不好说了。

我们将对表 2.4 中的几个选项进行进一步的说明：

- **hwconf** 包含了所有的硬件信息。Kudzu 服务检查它来看看用户是否已经添加或者删除了任何硬件，并询问用户对所做的改动采取什么措施。用户很可能想在一个工作系统上禁用这个服务，因为只要这个服务检测硬件配置有了变化，它都会延迟引导过程，造成为每个硬件变化额外多等 30s 以上的时间。
- **network-scripts** 目录包含与网络配置有关的补充信息。用户需要改动的惟一地方是名为 **ifcfg-*interface*** 的文件。例如，**network-scripts/ifcft-eth0** 包含接口 eth0 的配置参数。它设置接口的 IP 地址和连网参数。有关配置网络接口的更多信息请参见第 12.7 节。
- **sendmail** 文件包含两个变量：DAEMON 和 QUEUE。如果 DAEMON 变量设置为 yes，那么系统将在系统引导时就以守护进程模式（**-bd**）启动 **sendmail**。QUEUE 告诉 **sendmail**，在两次邮件排队操作（**-q**）之间间隔多长时间；默认是一小时。

2.5.3 SUSE 的启动脚本



虽然 SUSE 的启动脚本类似于 RHEL 和 Fedora 的启动脚本，但是 SUSE 的启动脚本确实是它比其他 Linux 变体显得耀眼的一个领域。SUSE 的脚本组织得不错，健壮可靠，而且有很好的文档说明。维护操作系统这一部分的人应该得到褒奖。

[2] 我们曾经有一个损坏的键映射（keymap）文件，由于即便是在单用户模式下也装载键映射，因此单用户模式就没有办法使用了。设置 **init=/bin/sh** 是引导系统到可以使用的单用户状态并解决这个问题的惟一办法。这个办法在其他一些情况下也是一个有用的窍门。

www.TopSage.com

大家网 TopSage.com

同Red Hat和Fedora系统里的情况一样，**init**在每个运行级都要把新运行级作为参数来调用**/etc/init.d/rc** 脚本。针对软件的脚本都在**/etc/init.d** 目录下，它们的配置文件则在**/etc/sysconfig** 目录下。**/etc/init.d/README** 里可以找到对 SUSE 引导过程很好的介绍。

虽然 SUSE 和 RHEL/Fedora 都把它们的引导配置文件放在了**/etc/sysconfig** 目录下，但是在这个目录下的具体文件则大有不同。（首先，SUSE 的文件一般都有很好的文档说明。）通过设置 shell 的环境变量就能调用配置选项，而这些变量接下来供**/etc/init.d** 里的脚本访问。有些子系统比别的子系统需要更多的配置，那些需要多个配置文件的子系统有专门的子目录，比如 **sysconfig/network** 目录。

windowmanager 文件是 sysconfig 目录下的一个典型例子：

```
## Path:          Desktop/Window manager
## Description:
## Type:          string(kde,fvwm,gnome,windowmaker)
## Default:       kde
## Config:        profiles,kde,susewm
#
# Here you can set the default window manager (kde, fvwm, ...)
# changes here require at least a re-login
DEFAULT_WM="kde"

## Type:          yesno
## Default:       yes
#
# install the SUSE extension for new users
# (theme and additional functions)
#
INSTALL_DESKTOP_EXTENSIONS="yes"
```

每个变量前面都有 YaST[3]可以读懂的配置信息，还有对这个变量用途的详细说明。例如，在**windowmanager** 文件里，DEFAULT_WM 这个变量设置了由 X 使用的桌面窗口管理器。

SUSE 在**/etc/sysconfig/network** 目录下的网络配置文件组织得格外不错。这个目录既包含全局配置文件（设置与所有网络接口有关的配置选项），也有特定于网络的文件。例如，**network/routes** 这个文件保存有全局的路由信息。在一个典型安装的 SUSE 系统上，它的内容类似下面的情况：

```
# Destination Dummy/Gateway  Netmask  Device
default       192.168.10.254 0.0.0.0  eth0
```

仅当一个特定网络接口启动且正在运行的时候应该出现的路由，在一个叫做 **ifroute-*ifname*** 的文件里指定。例如，对于名为 eth1 的接口，这个文件就成了 **ifroute-eth1**，其内容可以是：

```
# Destination    Dummy/Gateway   Netmask   Device
10.10.0.0/24     10.10.0.254
```

如果愿意，Netmask 和 Device 也能指定，但是启动脚本会推断出正确的值。

SUSE 也带一个管理启动脚本的 **chkconfig** 命令。它和 Red Hat 提供的版本完全不同，不过它仍然是一个有效的工具，在偏好手工管理脚本的情况下应该使用它。

不论选择使用 YaST 还是 **chkconfig**，或者手工维护启动脚本，浏览**/etc/sysconfig** 目录并且琢磨一下它的内容将会是一个好主意。

下面是一个典型的 SUSE 引导会话的例子：

[3] YaST 是 SUSE 特有的一种图形化配置工具，它可以维护一个 SUSE 系统许多方面的配置。参考第 11 章了解更多信息。

www.TopSage.com

```
[kernel information]
INIT: version 2.85 booting
System Boot Control: Running /etc/init.d/boot
Mounting /proc filesystem                                    done
Mounting sysfs on /sys                                       done
Mounting /dev/pts                                            done
Boot logging started on /dev/tty1(/dev/console) at Tue Mar 29 14:04:12 2005
Mounting shared memory FS on /dev/sh                         done
Activating swap-devices in /etc/fstab...
Adding 1052248k swap on /dev/hda2. Priority:42 extents:1     done
Checking root file system...
...
```

2.5.4 Debian 和 Ubuntu 的启动脚本



如果说 SUSE 是对管理启动脚本有良好设计、良好执行规划的典型，那么 Debian 却正好相反。Debian 的脚本脆弱、缺少文档而且还令人难以置信地不一致。真糟糕，在这一块，设置脚本的方式缺乏标准似乎造成了混乱。

在每个运行级中，**init** 都以新的运行级作为参数来调用脚本**/etc/init.d/rc**。每个脚本负责找到它自己的配置信息，这些信息可能是在**/etc**、**/etc/default**、以及**/etc** 下其他子目录里，或者在脚本自身中的什么地方。

如果用户正在查找系统的主机名，那它保存在**/etc/hostname** 里，供脚本**/etc/init.d/hostname.sh** 读取。网络接口和默认网关的参数保存在**/etc/network/interfaces** 里，由从**/etc/init.d/networking** 里调用的 **ifup** 命令读取。有些网络选项也会在**/etc/network/options** 中设置。

Debian 和 Utuntu 有一种神秘的启动脚本管理程序，其形式为 **update-rc.d**。虽然它的 man 手册警告说不要以交互的方式使用它，但是我们发现它虽然有点儿不友好，但还是能用的，可以替代 **chkconfig**。例如，要在运行级 2、3、4 和 5 都启动 **sshd**，而在运行级 0、1 和 6 停止它，就可以用：

```
$ sudo /usr/sbin/update-rc.d sshd start 0123 stop 456
```

2.6 重新引导和关机

Linux 的文件系统缓冲一般在内存中变化，只是偶尔才把它们写回磁盘。这种方案使得磁盘 I/O 速度更快，但是当系统被粗暴地中止时，这种方案也更容易使文件系统丢失数据。

传统的 UNIX 和 Linux 机器要非常小心地处理关机方式才行。现代的系统已经变得不那么敏感了（尤其是当用户使用一个像 ext3fs 这样强健的文件系统时），但有可能的话，妥善关机总是一个好主意。不恰当的关机可能产生隐蔽的小问题，从而导致重大灾难事故。

在面向消费级用户的操作系统上，重新引导操作系统是适合处理几乎任何故障的第一课。而在 Linux 系统上，最好首先考虑清楚，然后再重新引导机器。Linux 的故障更微妙、更复杂一些，因此，盲目地重新引导系统只会在很小比例的场合下起作用。还有，Linux 系统引导所花费的时间很长，这对于多用户来说是不太方便的。

当添加一种新硬件或现有硬件出现问题甚至不能够复位时，通常需要重新引导系统。如果所修改的配置文件只有在引导过程中才使用的话，那么，要让改动起作用，也必须重新引导系统。如果系统出现故障，使得用户不能够登录进入系统去正确地诊断故障，这时显然除了重新引导系统以外别无选择。

只要修改启动脚本，就应该重新引导系统，以确保系统能够成功地运行起来。如果时隔几个星期

www.TopSage.com

之后才发现问题，那么就不太可能回想起最近所做修改的细节。

引导系统本质上只能用一种方法来完成，与此不同的是，关闭或重新引导系统有许多种方法。这些方法如下：

- 关闭电源；
- 使用 **shutdown** 命令；
- 使用 **halt** 和 **reboot** 命令；
- 使用 **telinit** 来改变 **init** 的运行级别；
- 使用 **poweroff** 命令告诉系统关闭电源。

2.6.1 关闭电源

即使是在桌面系统上，关闭电源也不是关闭系统的一个好方法。这样做有可能丢失数据和破坏文件系统。

许多机器有一个“软电源开关”，当用户按下电源按钮时，机器实际上运行一串命令来执行关闭系统的正确操作。如果不清楚自己的机器是否提供这个特性，那么不要按下电源按钮来证明这一点！最好是自己去执行这一串关闭系统的操作。

不过，这并不是说绝对不可以关闭电源。在发生紧急情况时，如果没有足够的时间来妥善关机的话，那么关闭电源也是可以的。老式机房经常有一个“意外事件处理按钮”可以立即关闭所有设备的电源。我们的系统管理员曾经让一个踢歪了的足球触动了那个开关。

2.6.2 shutdown：停止系统的妥善方式

shutdown 是停止或重新引导系统，或者返回到单用户模式的最安全、考虑最周到的、最彻底的方式。

可以让 **shutdown** 在开始关闭系统以前等待一定的时间。在这个等待的过程中，**shutdown** 以越来越短的时间间隔发送一些消息给已经登录的用户，警告用户关闭系统的时间即将到来。默认情况下，这些警告消息只是说系统即将关闭并给出离关闭系统还剩的时间，管理员也可以提供自己的简短消息。消息应该说明为什么要关闭系统、并应该估计多长时间以后用户可以再次登录进入系统（比如，上午 11:00 恢复）。在大多数系统上，立即执行 **shutdown** 时，用户是不能够登录进入系统的，但如果管理员设定自己的消息，那么用户能够看到消息，从而明白是怎么回事。

shutdown 让管理员指定当该任务完成后，机器是应该停机（**-h**），还是重新引导（**-r**）。还可以指定在重新引导以后是强制使用 **fsck** 命令检查磁盘（**-F**），还是不检查（**-f**）。在默认情况下，只要文件系统是正确卸载的，Linux 系统会自动跳过 **fsck** 检查。

例如，一条提醒用户上午 9:30 实施预定的维护而停止系统运行的 **shutdown** 命令如下：

```
# shutdown -h 09:30 "Going down for scheduled maintenance. Expected
   downtime is 1 hour"
```

还可以指定相对的关机时间。例如，下面的命令将在它开始运行 15 分钟之后关机：

```
# shutdown -h +15 "Going down for emergency disk repair."
```

2.6.3 halt：关闭系统的更简单方式

halt 命令执行关闭系统所需要的基本任务。它可以被 **shutdown -h** 调用，也可以单独使用。**halt**

www.TopSage.com

记录关机的情况，终止那些非必需的进程，执行 **sync** 系统调用（被 **sync** 命令调用并与 **sync** 命令等效），等待文件系统写操作完成，然后停止内核。

halt -n 不执行 **sync** 调用。在 **fsck** 修好了根分区之后会用到这条命令。如果 **fsck** 没有使用**-n** 选项，那么内核可能会用内存中缓存的超级块的老版本覆盖 **fsck** 的修改。

2.6.4 reboot：快速的重新启动

reboot 几乎跟 **halt** 完全一样，只不过它是让机器重新启动，而 **halt** 是让机器停止。**reboot** 由 **shutdown -r** 来调用。与 **halt** 类似，它也支持**-n** 参数。

2.6.5 telinit:改变 init 的运行级

使用 **telinit** 可以指引 **init** 进入特定的运行级。例如：

```
# telinit 1
```

让系统进入单用户模式。

在使用 **telinit** 的时候，不会得到像 **shutdown** 那样友好的警告消息或者善意的等候时间，所以在大多数情况下，可能都要避免用到它。对于测试对 **inittab** 文件的修改来说，**telinit** 是最有用处的。

2.6.6 Poweroff：请 Linux 关闭电源

poweroff 命令等价于 **halt**，不同之处是在 Linux 关闭后 **poweroff** 可以向电源管理系统（在有这项功能的系统上）发送一则请求来关闭系统的主电源。这项功能易于实现远程关机（例如，在一次供电故障期间）。

遗憾的是，没有对应的 **poweron** 命令。这一疏忽的原因留给读者作为练习。

2.7 习题

E2.1 为什么在安装 LILO 引导加载程序之前先运行 **lilo -t** 很重要？该怎样引导一个名字不是 **vmlinuz** 的内核？

E2.2 为什么不应该用计算机机箱上的电源开关来关闭 Linux 系统？有什么别的替代办法吗？

E2.3 用 GRUB 的命令行引导一个 **grub.conf** 文件中没有的内核。

★E2.4 试述运行级的概念。列出 Linux 中定义的运行级，并简短描述一下每个运行级。运行级 1 和 S 之间有什么关系？

★E2.5 编写一个启动脚本，启动名为“foo”的守护进程（**/usr/local/sbin/foo**），它是一项网络服务。说明如何把它融入系统，在引导时刻自动启动。

★E2.6 取得并安装由 Dan Farmer 和 Wietse Venema 所写的程序 **mactime**（它是 TCT 工具集的一部分）。运行 **mactime**，创建一个与系统文件相关联的时间戳初始数据库。重启机器。再次运行 **mactime**，判断因为引导机器而被修改的文件有哪些。哪些文件访问了而没被修改？（需要 root 访问权限）

★★E2.7 如果一个系统处于运行级 4，而用户运行了命令 **telinit 1**，**init** 会采取哪些步骤？命

www.TopSage.com

令最终的结果将会是什么？

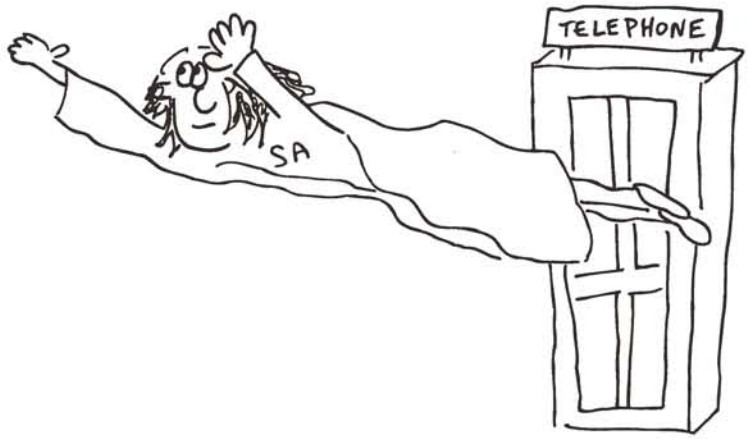
★★E2.8 绘制一张依赖关系图，画出在您的 Linux 系统上启动别的守护进程之前，必须启动哪些守护进程？

★★E2.9 顺序列出创建一个能工作的多 OS 系统——包含 Linux 和 Windows——所用到的步骤。使用 GRUB 和 Windows 的引导加载程序。

www.TopSage.com

大家网
TopSage.com

第 3 章　超级用户的权力



Linux 系统上的每个文件和进程都属于特定的用户账号。如果没有得到属主的许可，其他用户就不能访问这些对象，因此，这条约定有助于防止用户彼此之间不管是有意还是无意的错误行为。

系统文件和进程属于一个称为 root 的虚拟用户，也叫做超级用户。与任何其他账号一样，root 拥有的东西也受到保护，不受来自其他用户的干扰。要进行系统管理上的改动，需要使用本章中所描述的访问 root 账号的各种方法中的一种。

root 账号具有几个“神奇”特性。root 可以充当任何文件或进程的属主，还可以执行一些特殊操作，而其他用户是无权执行这些操作的。这个账号的权限非常大，但如果是没有经过训练的用户或者是心怀恶意的用户使用这个账号的话，将会非常危险。

本章介绍管理员使用超级用户身份的基础知识。第 20 章介绍怎样避免出现不必要和令人为难的由他人使用超级用户身份的情况。第 30 章介绍相关的政策和策略方面的内容。

3.1　文件和进程的归属关系

每个文件都同时有一个属主（owner）和一个“属组（group owner）”。文件的属主拥有一种不和系统上任何用户共享的特殊权限，那就是修改文件访问权限的能力。具体而言，就是属主可以严格地设置文件的访问权限，让其他任何人都不能够访问这个文件[1]。我们将在第 5 章中讨论文件权限。

尽管文件的属主总是一个人，但许多人都可以成为文件的属组成员，只要他们

[1] 事实上，访问权限可以设置严格到即使是文件的属主都不能够访问它。

www.TopSage.com

全部是一个 Linux 组的成员即可。组原本在**/etc/group** 文件中进行定义，但是现在用户组的信息保存在网络中的 NIS 或者 LDAP 服务器上的情形更为常见，参考第 17 章的内容了解详情。有关组的更多信息请参见 6.1 节。

文件的属主指定属组成员可以对文件进行哪些操作。这种机制能让同一个项目的成员之间共享文件。例如，我们使用了一个组来控制对 web 站点 www.admin.com 源文件的访问。

一个文件的这两种归属关系都可以用 **ls -l** *filename* 命令来了解，例如：

```
$ ls -l /staff/scott/todo
-rw-------  1 scott staff 1258 Jun 4 18:15  /staff/scott/todo
```

这个文件的属主是用户“scott”，属组是“staff”组。

Linux 实际上用数字来确定属主和属组，而不是用它们的文字名称。在最基本的情况下，用户标识号（user identification numbers，简记为 UID）被映射到**/etc/passwd** 文件中的用户名上，而组标识号（group identification numbers，GID）被映射到**/etc/group** 文件中的组名上。定义与 UID 和 GID 相对应的文字名称只是为了方便系统的现实用户使用。当诸如 **ls** 这样的命令要以人们可以阅读的格式来显示归属关系信息时，这些命令就必须在相应的文件或数据库中查出每个名字。

一个进程的属主可以向该进程发送信号（参见 4.3 节），还可以减小（降低）该进程的调度优先级。进程实际上至少有 7 个与之相关的标识：一个真实（real）UID、一个有效（effective）UID 和一个保存（saved）UID；一个真实 GID、一个有效 GID 和一个保存 GID；在 Linux 下，还有一个“文件系统（filesystem）UID”，它只用来确定文件的访问权限。概括地说，“真实”ID 用作记账，而“有效”ID 用来确定访问权限。一般情况下，真实 ID 和有效 ID 是相同的。

保存 ID 没有直接作用。它们能让程序“记下”一个不活动的 ID 供以后使用，减少需要用过大特权的机会。文件系统 ID 一般按 NFS 的实现细节来解释，通常和有效 ID 相同。

尽管通常对于一个进程来说不太可能会改变它自己的归属关系状态，但在一种特殊情形下，可以修改进程的有效 UID 和有效 GID。当执行设置了“setuid”或“setgid”权限位的命令时，所得到的进程其有效 UID 或 GID 可以设置为包含该程序的映像文件的 UID 或 GID，而不是设置为运行该命令的用户的 UID 和 GID。这样一来，用户的权限就为此次执行该特定命令而得到“提升”。有关权限位的更多信息，请参见第 5.5 节。

利用 Linux 的 setuid 功能，普通用户可以以一种受限的并被严格控制的方式来使用 root 账号去运行程序。例如，用户运行用来修改其登录口令的 **passwd** 命令就是一个 setuid 的程序。它以一种严格限定的方式去修改**/etc/shadow**（或者**/etc/passwd**）文件，然后退出执行。当然，即使是这类受限任务也有着被滥用的潜在可能，因此 **passwd** 命令在同意执行所请求的修改以前，要求用户证实他们知道当前账号的口令。

3.2 超级用户

root 账号在定义上的特征是它的 UID 为 0。Linux 并没有不许用户修改这个账号上的用户名，也没有不许创建 UID 为 0 的其他账号，但这两种做法都是非常不好的。像这样的修改有可能无意中破坏系统的安全性。同时，当其他人不得不适应以这种奇怪方式配置的系统时，也容易给他们带来混乱。

传统的 UNIX 允许超级用户（也就是有效 UID 为 0 的任何进程）在任何文件或进程上执行正当的操作[2]。此外，有些系统调用（对内核的请求）只能够由超级用户来执行，这类受限操作的一些例

[2] 这里的“正当”是一个重要的、意义含糊的词。某些操作（例如，执行没有设置执行许可位的文件）即使对于超级用户也是被禁止的。

www.TopSage.com

子有：

- 采用 **chroot** 命令来改变进程的根目录；
- 创建设备文件；
- 设置系统时钟；
- 提高资源使用率的限度和进程的优先级[3]；
- 设置系统的主机名称；
- 配置网络接口；
- 打开特权网络端口（那些编号小于 1024 的端口）；
- 关闭系统。

例如，超级用户有权改变属主为 root 的进程的 UID 和 GID。**login** 程序及其采用窗口形式的等价程序都是这样的例子，这个在用户登录到系统时提示用户输入口令的进程起初就是以 root 权限运行的。如果所输入的用户名和口令是合法的话，login 就把它的 UID 和 GID 改变成为该用户的 UID 和 GID 并启动该用户的用户环境。一旦一个以 root 权限执行的进程已经改变了它的归属关系而成为一个普通的用户进程，那么它就不能再恢复它以前的特权状态了。

Linux 系统从理论上说也能把 root 账号的特权按照 POSIX 标准中有关“权力（capability）”的规定再做细分。出于各种原因，其中也包括当前实现存在的问题，这项功能并没有其最初体现的那样有帮助，对于系统管理员来说也没有那么适合使用。有关权力的更多介绍，请参见 20.6 节。

3.3 选择 root 的口令

root 口令的长度至少应为 8 个字符，7 个字符长度的口令其实非常容易被破译。在有些使用 DES 口令的系统上，使用长度超过 8 个字符的口令是没有什么用处的，因为只有前 8 个字符有意义。请参见 6.1 节了解使用 MD5 口令的方法，这种口令的长度可以超过 8 个字符。参考第 20 章了解有关口令破解的更多知识。

让所选的 root 口令不那么容易被猜中或被反复试探找到，这是很重要的。从理论上来说，最安全的口令类型是由字母、标点符号和数字组成的随机序列。但是，因为这种类型的口令很难记住，并且一般来说输入也比较费劲，如果管理员要用笔写下才不会忘记，或者键入这种口令的速度比较慢，那么这未必就是最安全的口令。

直到最近，由两个随机选择的单词中间加一个标点符号所组成的口令还是在安全性和易记性之间比较好的折衷方案。遗憾的是，目前这类口令也可以被很快破译了，所以我们不提议大家采用这种方式的口令。

现在，我们建议您通过摘取“出人意料的废话（shocking nonsense）”中的短语来构成 root 的口令，“shocking nonsense”是 Grady Ward 在 PGP Passphrase FAQ（PGP 口令词的 FAQ）的一个较早版本中定义的：

“shocking nonsense”意味着组成这样一个短语或句子，在用户的文化中既没有什么意义，又是出人意料的（shocking）。也就是说，它包含了粗俗恶心的、种族主义的、不可能的或者其他类似的极端思想。因为口令究其本质来说，是从来不会给其他任何人看的，因而口令的意义也不会触犯别人，所以允许采用这项技术。

“出人意料的废话”不太可能在别处重复，因为它并没有描述一个有可能被其他人意外地重新发现的事实。对于创建者本人来说，突如其来的感情化的想法是很难忘记的。例如，“Mollusks peck my

[3] 截至 2.6.12 版的内核，一种新的资源限度能让除了超级用户之外的其他用户在系统管理员允许的情况下提高进程的优先级。

www.TopSage.com

galloping genitals”，读者无疑可以为自己创作出更多的或更有趣的例子。

通过只记住每个单词的第一个字母或者其他类似的转换方法，就可以把这样的短语缩减成为 8 个字符长度的口令。如果其中还包括数字、标点符号和大写字母，那么口令的安全性将大幅提高。

您应该改变 root 的口令：

- 至少每 3 个月左右；
- 每次当有个知道口令的人离开您的工作站点时；
- 在认为安全性可能已经受到威胁的任何时候；
- 当某一天，您没有料到晚上的聚会很累以致于第二天早上会忘记口令时。

3.4 成为 root 用户

root 不过是另一个用户而已，所以可以直接以 root 账号登录进入系统。然而，这种做法显得非常糟糕。首先，这样做没有留下以 root 身份执行的操作的记录。当管理员意识到自己昨天夜里凌晨 3:00 时损坏了某项内容，但又记不起来自己当时到底改变了什么时，这可就不妙了。当访问没有经过授权，而要试图找出入侵者对系统到底做了什么操作时，情况可能更糟。另外一个不利因素是以 root 身份登录进入系统的做法没有留下当时是谁真正做这项工作的记录。如果有几个人都有访问 root 账号的权限，那么将不能够区分到底是谁在什么时候使用了 root 账号。

基于上述原因，大多数系统禁止 root 在终端上以及通过网络来登录系统，或者说除了系统控制台之外的其他任何地方都禁止 root 登录[4]。我们建议您使用这项特性，请参见 20.9 节以确定在您的特定系统上需要编辑哪个文件来配置这一功能。

3.4.1 su：替换用户身份

访问 root 账号稍好一些的方式是使用 **su** 命令。如果不带任何参数来调用 **su** 命令，那么它会提示用户输入 root 的口令，然后启动一个 root shell。这个 shell 的特权一直到 shell 终止（通过<Control-D>或 **exit** 命令来终止）以前都保持有效。**su** 不记录以 root 身份执行了哪些命令，但它确实会创建一条日志记录来说明是谁在什么时候变成了 root。

su 命令也可以替代除了 root 以外的其他用户身份。有时候，重现或者调试用户问题的惟一方法就是 **su** 为该用户的账号，以便能够重现发生问题的环境。

如果知道某人的口令，那么通过执行 **su** *username* 就能够直接访问这个人的账号。与 **su** 到 root 账号的情况一样，此时也会提示用户输入 *username* 的口令。你也可以首先 su 到 root 账号，然后再 su 到其他账号，root 账号不需提供密码就可以 su 到其他任何账号。

最好是键入 **su** 命令的完整路径（例如，**/bin/su**），而不是依靠 shell 为你找到这个命令。这将在一定程度上防范已经溜入到搜索路径下的名为 **su** 的程序，它们带有想要获取口令的企图[5]。

3.4.2 sudo：受限的 su

由于超级用户账号的特权不可分割（至少不能随意分割），所以很难既给某个人完成一件任务的

[4] Ubuntu Linux 的限制更严。在默认情况下，系统没有合法的 root 口令，要用 sudo 来获得 root 访问权，本节后面会详细介绍。

[5] 由于相同的原因，我们强烈建议您在 shell 的搜索路径中不要包括“.”（当前目录）。虽然这样做会带来方便，但是该配置很容易意外地运行用户或入侵者作为陷阱留下的系统命令的特殊版本。自然，本忠告对于 root 用户加倍重要。

www.TopSage.com

权力（例如，备份）而又不给这个人自由运行系统的权力。而且如果root账号由几个管理员使用，那么对于是谁正在使用这个账号或者谁使用这个账号做了什么事情就只有非常模糊的认识了。

针对这些问题，最广泛采用的解决方案是使用一个叫做**sudo**的程序，这个程序目前由Todd Miller维护。我们所举的Linux发行版本都提供了这个软件。

sudo采用命令行作为参数，然后以root身份（或另外一个受限用户）执行。**sudo**读取文件**/etc/sudoers**，这个文件列出了授权使用**sudo**的人以及允许他们在每台主机上运行的命令。如果提供给**sudo**的命令允许运行，那么**sudo**就提示输入这个用户自己的口令并执行命令。

使用**sudo**的用户随后可以执行其他一些**sudo**命令而不用再输入口令，除非在超过5min（时间长度是可以配置的）的时间内该用户没有再使用**sudo**。这段超时时间可以作为一种最友善的保护，防止具有**sudo**特权的用户由于不注意而长时间地离开终端所造成的安全隐患。

sudo保存有一个日志，它记录执行的命令行、命令行执行的主机、请求执行命令行的人、运行命令行的目录以及命令行被调用的时间。这些信息可以通过syslog进行记录或者被放入选定的文件中。我们推荐使用syslog把日志项转发给一个安全的中央主机。

用户randy执行**sudo /bin/cat/etc/sudoers**命令会产生类似下面的一条日志记录：

```
Dec 7 10:57:19 tigger sudo: randy: TTY=ttyp0 ; PWD=/tigger/users/randy;
    USER=root ; COMMAND=/bin/cat /etc/sudoers
```

文件**sudoers**的设计做到了仅用一个版本就可以立即用在许多不同的主机上。下面是一个典型的例子：

```
# Define aliases for machines in CS & Physics departments
Host_Alias    CS = tigger, anchor, piper, moet, sigi
Host_Alias    PHYSICS = eprince, pprince, icarus

# Define collections of commands
Cmnd_Alias  DUMP = /sbin/dump, /sbin/restore
Cmnd_Alias  PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias  SHELLS = /bin/sh, /bin/tcsh, /bin/bash, /bin/ash, /bin/bsh

# Permissions
mark, ed      PHYSICS = ALL
herb          CS = /usr/sbin/tcpdump : PHYSICS = (operator) DUMP
lynda         ALL = (ALL) ALL, !SHELLS
%wheel        ALL, !PHYSICS = NOPASSWD: PRINTING
```

前5个非注释行定义了主机组和命令组，这些组供这个文件后面的权限说明来引用。这个列表可以原封不动地逐字包括在说明中，但使用别名会让**soduers**文件更容易阅读和理解，还能让文件以后更容易更新。还可以为用户集合或以其身份运行命令的用户集合定义别名。

每一行权限说明包括的信息如下：

- 这一行所适用的用户；
- 应该注意这一行的主机；
- 指定用户可以运行的命令；
- 可以以其身份执行命令的用户。

第1行权限说明适用于PHYSICS组机器（eprince、pprince和icarus）上的用户mark和ed。**sudo**内建的命令别名ALL允许他们运行任何命令。由于在小括号中没有指定用户列表，所以**sudo**将只以root身份运行命令。

第2行权限说明允许用户herb在CS组机器上运行**tcpdump**，并在PHYSICS组机器上运行与dump有关的命令。不过，dump命令只能够以operator的身份而不能以root身份运行。用户herb键入的实

www.TopSage.com

大家网 TopSage.com

际命令行则类似下面这样：

```
$ sudo -u operator /sbin/dump 0u /dev/hda2
```

用户 lynda 能够以任何用户的身份在任何机器上运行命令，但不能够运行几个普通的 shell，难道这就意味着 lynda 不能够得到一个 **root shell**？当然不是：

```
$ cp -p /bin/bash /tmp/bash
$ sudo /tmp/bash
```

一般说来，任何试图设定“除……以外的所有命令”的做法都注定要失败，至少在技术角度来看是这样。但是，按照这种方式建立一个 **sudoers** 文件，作为不鼓励使用 root shell 的提醒信息仍然是值得的。这种做法可能会避免用户不经意的使用。

最后一行允许组“wheel”中的用户以 root 身份在除了 eprince、pprince 和 icarus 之外的任何机器上运行 **lpc** 和 **lprm**。而且运行这些命令不需要任何口令。

注意，**/etc/sudoers** 中的命令采用完整路径来指定，这样做是为了防止人们以 root 身份去执行他们自己的程序和脚本。尽管上面没有给出例子，但可以为每个命令指定允许的参数。事实上，**sudoers** 文件的功能相当强大，这个简单的配置只涉及到它的一点点功能。

要修改**/etc/sudoers**，请使用 **visudo** 命令，这个命令检查以确保没有其他人正在编辑这个文件，接着调用一个编辑器编辑该文件，然后先验证文件编辑修改后的语法无误，再安装它。最后这一步骤尤其重要，因为无效的 **sudoers** 文件可能会不让用户再用 **sudo** 去修复它。

使用 **sudo** 具有以下一些好处：

- 由于有命令日志，因而极大地提高了安全审计的能力；
- 操作员不需要不受限制的 root 特权就能够完成许多任务；
- 真正的 root 口令可以只让一两个人知道；
- 使用 **sudo** 比运行 **su** 或者以 root 身份登录进入系统要快；
- 不需要改变 root 的口令就能够收回一些特权；
- 维护一个具有 root 特权的所有用户的规范列表；
- 不经意遗留 root shell 的概率降低了许多；
- 可以使用单个文件来控制对整个网络的访问权限。

sudo 当然也有一些不足之处。其中最糟糕的是，如果突破了能执行 sudo 命令的个人账号的安全防线，就等同于突破了 root 账号本身的安全防线。对付这种威胁除了提醒能执行 sodo 命令的用户在他们暂时成为 root 时保护好自己的账号之外，就没有其他什么措施了。也可以定期用 John the Ripper 程序尝试破解一下 sudoer 的口令，确保用户选择了良好的口令。参考 20.10 节了解有关 John the Ripper 程序的更多知识。

sudo 的命令日志机制可以用一些技巧暗中破坏，比如在允许执行的程序中使用 shell 的转义字符，或者如果允许 **sudo sh** 和 **sudo su** 的话，则利用它们来破坏。

3.5 其他的伪用户

在内核看来，root 是具有特殊地位的惟一用户，但系统还定义了另外几个伪用户。惯用的做法是用星号来替代这些特殊用户在**/etc/passwd** 中被加密的口令字段，从而让这些用户的账号不能用来登录进入系统。

3.5.1 bin：系统命令的老属主

在一些比较老的 UNIX 系统上，bin 用户是包含系统命令的那些目录的属主，还是大多数命令本身的属主。如今这种账号经常被看作是多余的(或者甚至可能有些不安全)，因此现代系统(包括 Linux)通常就只使用 root 账号了。另一方面，既然 bin 账号是“标准”账号，所以还真的不能废除它。

3.5.2 daemon：无特权系统软件的属主

有些文件和进程是操作系统的一部分，但不需要由 root 做其属主，这样的文件和进程就给了 daemon。其中的道理是，这种约定能有助于避免采用 root 做属主所带来的危险。出于类似的原因，还有一个叫做“daemon”的组。和 bin 账号类似，大多数 Linux 发行版本也不怎么用 daemon 账号。

3.5.3 nobody：普通 NFS 用户

网络文件系统（Network File System，NFS）在进行文件共享的时候，使用 nobody 账号代表其他系统上的 root 用户。为了去掉远程 root 用户的特权，远程 UID 为 0 的用户必须被映射成本地 UID 0 之外的某个用户。nobody 账号就充当了这些远程 root 用户的一般替身。参考 16.1 节了解有关 nobody 账号的更多信息。

由于 nobody 账号应该代表一个普通的、权力相对来说比较小的用户，因此这个账号不应该拥有任何文件。如果 nobody 账号确实拥有文件的话，那么远程的 root 就可以控制这些文件。nobody 不应该拥有任何文件！

传统上给 nobody 用户的 UID 是-1 或者-2，Linux 内核仍然默认用 UID 65534（-2 的 16 位二进制补码）。有些发行版本给 nobody 指派一个小编号的 UID（例如，Red Hat 和 Fedora 使用 9），这样做比 65534 更合理，因为 UID 是 32 位的。惟一有问题的地方是 **exportfs** 命令似乎不去看 **passwd** 文件，所以必须用 **anonuid** 选项明确告诉它给 nobody 用一个不同的 UID。

3.6 习题

★E3.1 使用带有**-perm** 选项的 **find** 命令在您的系统上找到 5 个 setuid 文件。对于每个文件，试述为什么对于让命令正确发挥其功能来说，setuid 机制是必要的。

★E3.2 创建三个“出人意料的废话”式口令词，但是要保密。通过运行 **md5sum** 程序处理这三个口令词，报告结果。为什么公开 MD5 结果是安全的？

★E3.3 列举一系列修改某个用户口令的命令，展示如何掩盖修改的痕迹。假定您只有 **sudo** 的权利（允许使用除 shell 或者 **su** 之外的所有命令）。

★E3.4 为 **sudoers** 配置文件创建两项：

a）一项让用户 matt、adam 和 drew 负责在 printserver 这台机器上为打印机服务，消除夹纸，然后重新启动打印机的守护进程；

b）一项让 drew、smithgr 和 jimlane 终止学生机房里机器上的任务并重启机器。

★E3.5 安装 **sudo**，配置成让它把有关误用的邮件发给您。以本地用户名和机器名来测试上一个问题中的 **sudo** 配置项；验证 **sudo** 是否能正确地把日志写到 syslog。检查测试在 syslog 中生成的日志项(需要 root 访问权限，很可能还要调整一下**/etc/syslog.conf**)。

www.TopSage.com

第4章 进程控制



进程是 Linux 用来表示正在运行的程序的一种抽象概念。程序的内存使用、处理器时间和 I/O 资源就是通过这个对象进行管理和监视的。

Linux 和 UNIX 设计思想的一部分内容就是让尽可能多的工作在进程的上下文（context）中完成，而不是由内核专门来进行处理。系统进程和用户进程都遵守同样的规则，因此用户可以使用一套工具控制这两种进程。

4.1 进程的组成部分

一个进程由一个地址空间和内核内部的一组数据结构组成。地址空间是由内核标记出来供进程使用的一组内存页面[1]。它包含进程正在执行的代码和库、进程变量、进程堆栈以及在进程运行的同时内核所需要的各种其他信息。因为 Linux 是一种采用虚拟内存的系统，因此内存页面在地址空间中的位置和它在机器的物理内存或交换空间中的位置之间并不一定相关。

内核的内部数据结构记录了有关每个进程的各种信息，其中一些非常重要的信息有：

- 进程的地址空间映射；
- 进程的当前状态（睡眠状态、停止状态、可运行状态等）；
- 进程执行的优先级；
- 进程已用资源的信息；
- 进程已打开的文件和网络端口的信息；

[1] 页面是管理内存的单位，在 PC 上的页面大小通常是 4K。

www.TopSage.com

- 进程的信号掩码（一个记录，确定要封锁哪些信号）；
- 进程的属主。

这些属性有些可以由构成“线程组”的几个进程所共享，Linux 中的线程组和传统 UNIX 中的多线程进程类似。虽然它们共享同一个地址空间，但是一个线程组的成员们却有各自的执行权限和执行状态。实际上，系统管理员所关注的进程很少使用多线程执行机制，而且即便是那些采用了多线程机制的进程（比如 BIND 9 的 named），一般也不会要求管理员去关注线程那么细的粒度。

与进程有关的许多参数直接影响进程的执行：进程所获处理器的时间量、进程能够访问的文件等。在下面各小节中，我们将讨论系统管理员最感兴趣的那些参数的意义及其重要性。这些属性对于所有版本的 UNIX 和 Linux 来说都是相同的。

4.1.1 PID：进程的 ID 号

内核给每个进程分配一个独一无二的 ID 号。控制进程的大多数命令和系统调用需要用户指定 PID 来标识操作的目标。PID 按照创建进程的顺序来分配。

4.1.2 PPID：父 PID

Linux 没有提供创建新进程去运行某个特定程序的系统调用，现有进程必须克隆自身去创建一个新进程。克隆出的进程能够把它正在运行的那个程序替换成另一个不同程序。

当一个进程被克隆时，原来的进程就叫做父进程，而克隆出的副本则叫做子进程。进程的 PPID 属性就是克隆它的父进程的 PID[2]。

当遇到无法辨认（以及可能运行失常）的进程时，父进程 PID 就成了一项很有用的信息。回溯该进程的父进程，有助于了解它的目的和作用。

4.1.3 UID 和 EUID：真实的和有效的用户 ID

进程的 UID 就是其创建者的用户标识号，或者更确切地说，就是复制了父进程的 UID 值。通常，只允许创建者（也称为属主）和超级用户对进程进行操作。有关 UID 的更多信息，请参见 6.1 节。

EUID 是“有效（effective）”的用户 ID，这是一个额外的 UID，用来确定进程在任何给定的时刻对哪些资源和文件具有访问权限。对于大多数进程来说，UID 和 EUID 是一样的，例外的情况是 setuid 程序。

为什么同时采用 UID 和 EUID 呢？这只是因为保持标识和权限之间的区别很有用，还因为 setuid 程序可能不希望一直以扩大了的权限执行操作。可以设置和重置进程的有效 UID 以便启用或限制它所享有的额外权限。

Linux 还有一种“保存 UID（saved UID）”，它是进程刚开始执行时刻，进程 EUID 的副本。除非进程采取措施删除这个保存下来的 UID，否则它就留了下来，作为真实的或者有效的 UID 来用。因此，编写严谨的 setuid 程序可以让其大部执行操作与它的特殊权限无关，只有在需要额外特权的特定时刻才用上它们。

Linux 还定义了一种非标准的进程参数 FSUID，它控制着对文件系统权限的判断，但在内核之外并不常用。

这种多 UID 体系的用意相当巧妙。如果用户需要深入了解细节，那么有一个特别好的资源——由

[2] 至少最初是这样的。如果原来的父进程终止，那么 init（进程 1）就成为新的父进程，请参见 4.2 节。

www.TopSage.com

David A. Wheeler 维护的免费网络在线图书 *Secure Programming for Linux and Unix HOWTO*。该书可以从 www.dwheeler.com 得到。

4.1.4 GID 和 EGID：真实的和有效的组 ID

GID 就是进程的组标识号。EGID 与 GID 的关系跟 EUID 与 UID 的关系相同，因为它可以由一个 setgid 程序来“转换”。Linux 有一种保存（saved）GID，就像有保存 UID 一样。

进程的 GID 属性基本没什么用处了。从确定访问权限这样的目的来说，一个进程可以同时是多个组的成员。完整的组清单区分 GID 和 EGID 独立保存。决定访问权限一般要考虑 EGID 和追加的组清单，但不考虑 GID。有关组的更多内容请参见 6.1 节。

只有当一个进程要创建新文件的时候，GID 才会起作用。文件系统权限是如何设定的，决定了新的文件可能要采用创建该文件的进程的 GID。参考 5.4 节的内容了解更多资料。

4.1.5 谦让度

进程的调度优先级决定了它所接收到的 CPU 时间有多少。内核使用动态算法来计算优先级：它考虑一个进程近来已经消耗 CPU 的时间量以及考虑该进程已经等待运行的时间等因素。内核还会关注为管理目的而设置的值，这种值通常叫做“谦让值（nice value）”或“谦让度（niceness）”，之所以这么叫是因为它表明了管理员计划对待系统其他用户的友好程度。我们将在 4.6 节中详细介绍这个主题。

为了给响应快的应用提供更好的支持，Linux 向传统的 UNIX 调度模型中增加了“调度类（scheduling class）”的概念。目前有三种调度类，每个进程都属于一种调度类。遗憾的是，实时类（real-time class）既没有广泛使用，也没有很好地从命令行给予支持。系统进程都使用传统的基于谦让度的调度机制。我们在本书里只讨论标准的调度机制。参考 www.realtimelinuxfoundation.org 了解有关实时调度机制的更多知识。

4.1.6 控制终端

大多数不是守护进程（daemon）的进程都有一个与自己相关联的控制终端。控制终端决定了为标准输入、标准输出和标准错误通道而提供的默认链接。当用户从 shell 启动一个命令时，他的终端通常就成为该进程的控制终端。控制终端的概念还影响到信号的发布，有关信号的概念将在 4.3 节中讨论。

4.2 进程的生命周期

为了创建一个新进程，一个进程就会用系统调用 **fork** 来复制自身。**fork** 创建原进程的一个副本，这个副本在很大程度上与父进程相同。新进程拥有一个不同的 PID 和它自己的记账信息。

fork 具有能够返回两个不同值的特性。从子进程的角度来看，它返回 0。另一方面，对于父进程来说，则返回新创建的子进程的 PID。由于这两个进程在其他方面是相同的，所以它们都必须检查返回值来确定它们要扮演什么角色。

在 **fork** 以后，子进程经常使用 **exec** 族系统调用中的一个成员开始执行新的程序[3]。这些调用能改变进程正在执行的程序正文（text），并把数据和堆栈段重新设定到预先定义的初始状态。**exec** 各种不

[3] 实际上，这个家族除了一个成员之外，都是库例程，而不是系统调用。

www.TopSage.com

同形式的区别仅在于它们为新程序指定命令行参数和环境时采用的方式不同。

Linux 定义 **fork** 的另一种实现 **clone**。这个调用创建的进程组共享内存和 I/O 空间。这一特性与大多数版本的 UNIX 上有的多线程机制类似，但是执行的每个线程都表现为一个完整的进程而不是一个专门的“线程”。

当系统引导时，内核会自主创建并安装几个进程。其中最知名的就是 **init**，它的进程号总是 1。**init** 负责执行系统的启动脚本。除了内核创建的那几个进程以外，其他所有进程都是 **init** 的后代。有关 **init** 守护进程的更多信息，请参考第 2 章。

init 在进程管理中还担负着另外一个重要角色。当一个进程执行完毕时，它调用一个名为 **_exit** 的例程来通知内核它已经做好“消亡”的准备了。它提供一个退出码（一个整数）表明它准备退出的原因。按照惯例，0 用来表示正常的或者说“成功”的终止。

在允许进程完全消失以前，Linux 要求该进程的消亡得到其父进程的确认，父进程是通过调用 **wait** 来确认的。父进程接收到子进程退出码的一个副本（或者是一个通知，在子进程不是自愿退出的情况下，说明子进程被终止的原因），如果父进程愿意，它还可以得到子进程对资源使用情况的一个总结。

如果父进程比子进程的生命期长并能负责地调用 **wait** 清理死亡的进程，那么这种方案可以工作得很好。但是，如果父进程比子进程先消亡，那么内核则会意识到将来不会有 **wait** 调用处理子进程了，于是就调整子进程使它这个“孤儿”成为 **init** 的子进程。内核要求 **init** 接受这些“孤儿”进程，并在它们消亡的时候执行所需的 **wait** 调用清除它们。

4.3 信号

信号是进程级的中断请求。系统定义了大约 30 种不同种类的信号，使用这些信号的方式可以有以下几种：

- 作为一种通信手段在进程之间发送信号；
- 当键入特殊的按键时，例如<Control-C>和<Control-Z>，可以由终端驱动程序发送信号去终止、中断或挂起进程[4]；
- 可以由管理员（使用 **kill**）发送信号来达到各种结果；
- 当进程出错，例如出现被零除的错误时，可以由内核发送信号；
- 可以由内核发送信号，通知一个进程有某种“受关注的”条件出现，比如一个子进程死亡或者在 I/O 通道上有数据。

当收到信号时，可能发生两种情况之一。如果接收进程已经为这个特定的信号指派了一个信号处理例程，那么就使用和传递该信号的上下文有关的信息来调用这个信号处理例程。否则，内核代表该进程采取某种默认措施。根据信号的不同，采取的默认措施也不同。许多信号会终止进程，有些信号也会产成一次内存转储（译者注：core dump 就是一个进程的内存映像。它可以用来调试进程。core 代表老式计算机上用作主存储器的磁芯“magnetic core”。现代计算机已经不再使用磁芯存储器了，或许称之为 memory dump 会更恰当些，但是导线 core 一词却沿袭下来，而且人们常常把 core 同 kernel 这个词混淆，以为是内核或核心的同义词，但实际上两者的含义大相径庭）。

在一个程序内为信号指定信号处理例程的做法通常叫做“捕获”该信号。当信号处理例程完成时，执行流程从接收到信号的位置重新开始。

[4] 可以使用 **stty** 命令将<Control-Z>和<Control-C>的功能重新分配给其他键，但实际上很少有人这样做。在本章中，我们通过习惯上的绑定关系来引用它们。

www.TopSage.com

为了防止收到信号，程序可以要求忽略（ignore）信号或者封锁（block）信号。被忽略的信号被简单地丢弃，它对进程没有什么影响。被封锁的信号排队等待发送，但内核不要求进程对该信号采取措施，一直到该信号被明确地解除封锁为止。处理新的不被封锁的信号的信号处理器只被调用一次，即使在封锁接收的同时多次接收到该信号的情况下，也只调用一次。

表 4.1 列出了所有管理员都应该知道的信号。信号名称采用大写的约定来源于 C 语言的传统，有时候还可能看到信号名称带有前缀 SIG（例如，SIGHUP），也是出于类似原因。

表 4.1　　每个系统管理员都应该知道的信号[a]

#	名称	描述	默认	是否能捕获?	是否能封锁?	是否转储内存映像?
1	HUP	挂起	终止	是	是	否
2	INT	中断	终止	是	是	否
3	QUIT	退出	终止	是	是	是
9	KILL	杀死	终止	否	否	否
b	BUS	总线错误	终止	是	是	是
11	SEGV	段错误	终止	是	是	是
15	TERM	软件终止	终止	是	是	否
b	STOP	停止	停止	否	否	否
b	TSTP	键盘停止	停止	是	是	否
b	CONT	停止以后继续	忽略	是	否	否
b	WINCH	窗口改变	忽略	是	是	否
b	USR1	用户定义	终止	是	是	否
b	USR2	用户定义	终止	是	是	否

a. 执行 **bash** 的内置命令 **kill-l** 也能获得一份信号名称和编号的清单。
b. 随硬件体系结构的不同而异，请参见 **man 7 signal** 命令了解更多的专门信息。

还有其他一些信号没有在表 4.1 中列出来，其中大多数用来报告一些含糊的错误，例如“illegal instruction（非法指令）”。对这类信号的默认处理方式是终止其执行并产生一个内存转储。通常也允许捕获和封锁信号，因为有些程序可能足够聪明，先尝试清除导致错误的任何问题，再继续运行。

BUS 和 SEGV 信号也是错误信号。我们之所以把这两个信号包括在表中，是因为这两个信号非常常见：一个程序崩溃的次数中有 99%最终都可以归结为是这两种错误中的一种导致的。这两个信号本身并没有特定的诊断值，它们都表明有不正确地使用或访问内存的企图[5]。

KILL 和 STOP 信号不能够被捕获、封锁或忽略。KILL 信号清除接收到它的进程。STOP 信号挂起正在执行的进程，一直到该进程接收到 CONT 信号为止。CONT 信号可以被捕获和忽略，但不能够被封锁。

TSTP 信号是 STOP 信号的一个“软”版本，把它描述为对“停止”的请求是最合适不过的了。这是当用户在键盘上键入<Control-Z>时由终端驱动程序生成的信号。捕获到这个信号的程序通常清除它们的状态，然后给自己发送一个 STOP 信号来完成停止操作。另外，TSTP 可以被简单地忽略，以防止程序被来自键盘的输入所停止。

终端模拟器在其配置参数（例如虚拟终端中的行数）改变时，都将发送一个 WINCH 信号。这项约定可以让那些理解模拟器的程序（比如文本编辑器）根据相应的变化而自动对自身进行重新配置。

[5] 更具体地来说，是由于没有满足对齐要求，或者使用无意义的地址所引发的总线错误。“segmentation violation”表明出现了诸如向地址空间的只读部分写数据这样的保护错。

www.TopSage.com

如果不能够让窗口正确地改变大小，那么请确认 WINCH 是否正确地产生和发送了[6]。

信号 KILL、INT、TERM、HUP 和 QUIT 听起来似乎表示同一回事，但实际上它们的使用方法非常不同。遗憾的是，人们竟然为这些信号选择了那样含义模糊的术语，下面是确切的解释。

- KILL 不可以封锁，它在内核的层面上终止进程。进程实际上从来就不能够“接收”到这个信号。
- INT 是当用户键入<Control-C>时由终端驱动程序发送的信号。这是一个终止当前操作的请求。如果捕获了这个信号，一些简单的程序应该退出，或者只是让自己被杀死，这也是程序没有捕捉到这个信号时的默认处理方法。拥有命令行模式的那些程序应该停止它们正在做的事情，清除状态，并等待用户的再次输入。
- TERM 请求彻底终止某项执行操作。它期望接收进程清除自己的状态并退出。
- HUP 有两种常见的解释。第一种，它被许多守护进程理解为一个重置的请求。如果一个守护进程不用重新启动就能够重新读取它自己的配置文件并调整自己以适应变化，那么 HUP 信号通常可以用来触发这种行为。第二种，HUP 信号有时候由终端驱动程序生成，试图用来“清除”（也就是“杀死”）跟某个特定终端相连接的那些进程。例如，当一个终端会话结束时，或者当一个调制解调器连接被不经意地断开（因而得名为“挂断”）时，就可能出现这种情况。各种不同的系统在具体的处理细节上有些区别。C shell 系列（tcsh 等）中的 shell 通常让后台程序不受 HUP 信号的影响，这样，它们可以在用户注销后继续运行。Bourn 风格 shell（**ksh**、**bash** 等）的用户可以使用 **nohup** 命令来模仿这种行为。
- QUIT 与 TERM 类似，不同之处在于，如果没有被捕获的话，它的默认行为是产生一个内存转储。有一些程序把这个信号解释为其他意思。
- 信号 USR1 和 USR2 没有设定意义。程序可以按照自己需要的方式，任意使用这两个信号，例如，Apache web 服务器把信号 USR1 解释为一次要求妥善重启的请求。

4.4 KILL 和 KILLALL：发送信号

顾名思义，**kill** 命令最常见的用法是终止一个进程。**kill** 能够发送任何信号，但在默认情况下，它发送一个 TERM 信号。**kill** 可以被普通用户用在他们自己的进程上，或者被超级用户用在任何进程上。语法是：

```
kill [-signal] pid
```

这里的 *signal* 就是要发送信号的编号或符号名称（如表 4.1 所示），*pid* 就是目标进程的进程标识号。pid 为-1 会把这个信号广播给除了 **init** 以外的所有进程。

没有信号编号的 **kill** 命令不保证进程会被杀死，因为 TERM 信号可能被捕获、封锁或忽略。下面的命令：

```
kill -KILL pid
```

将“保证”进程的消亡，因为信号 9，即 KILL 不能够被捕获到。我们给“保证”加引号是因为进程的生命力有时候能够变得相当“旺盛”，以致于连 KILL 也不能够影响到它们（通常是由于有些退

[6] 这点说起来要比做起来容易。终端模拟程序（例如，xterm）、终端驱动程序和用户级命令在出现 SIGWINCH 信号时都可能要负有一定的责任。常见的问题有，只向一个终端的前台进程（而不是所有和这个终端关联的进程）发送信号，没有把终端大小的变化通过网络告诉远端的一台计算机。像 TELNET 和 SSH 这样的协议都能识别本地终端大小的变化，并且把这一信息告诉远端的主机。比较简单的协议（例如，直连的串口线）就做不到。

www.TopSage.com

化的 I/O 虚假锁定，例如等待已经停止旋转的磁盘）。重新启动系统通常是解决这些“不听话”的进程的惟一方法。

大多数 shell 都内建有遵循上述语法规则的 **kill** 实现。根据独立的 **kill** 命令的 man 手册页的说法，信号名称或者编号之前实际上都有**-s** 标志（例如，**kill -s HUP** *pid*）。但是因为有些 shell 不理解这种语法，我们建议读者还是坚持采用**-HUP** 的形式，独立的 **kill** 命令也理解它。那样一来，用户就不需要关心正在用的是哪一个版本的 **kill** 了。

如果用户不知道要发信号的进程的 PID，一般应该用 **ps** 命令查出来，**ps** 命令将在 4.7 节介绍。另一种做法是使用 **killall** 命令，该命令替用户查出进程的 PID。例如，要刷新守护进程 **xinetd** 的配置，可以运行

```
$ sudo killall -USR1 xinetd
```

注意，如果用户的输入匹配多个进程，那么就会把 **killall** 信号发给所有匹配的进程。

普通的 **kill** 命令实际上有类似的功能，但是在匹配命令名上似乎没有 **killall** 那么聪明。还是坚持用 **killall** 吧。

4.5 进程的状态

进程不会仅仅因为其存在就自动地具有获得 CPU 时间的资格。管理员需要注意 4 种最基本的执行状态，这些状态将在表 4.2 中列出。

表 4.2　　进程的状态

状　态	意　义
Runnable（可运行状态）	进程可以被执行
Sleeping（睡眠状态）	进程正在等待某些资源
Zombie（僵化状态）	进程试图消亡
Stopped（停止状态）	进程被挂起（不允许执行）

处于可运行状态的进程只要有 CPU 时间可用，就准备执行。处于这种状态的进程已经得到它需要的全部资源，而只是在等待获得 CPU 时间去处理它的数据。一旦进程执行了一个不能够立即完成的系统调用（例如请求读取文件的一部分），Linux 将把这个进程转入睡眠状态。

处于睡眠状态的进程等待特定的事件发生。交互式的 shell 和系统守护进程把它们的大多数时间花费在睡眠、等待终端输入或网络连接上。由于正在睡眠的进程被有效地阻塞，直到它的请求被满足为止，所以，除非它接收到一个信号，否则它将得不到 CPU 时间。

有些操作让进程进入到一种不能中断的睡眠状态。这种状态通常是临时性的，从 **ps** 命令的输出看不到（在 STAT 列用一个 D 来指示，参见 4.7 节）。不过，在个别不对劲儿的情形下，会让这种状态保持下去。最常见的原因是，以“hard”选项通过 NFS 安装文件系统时，服务器出现问题。因为处于不能中断睡眠状态的进程甚至不能被唤醒去处理一个信号，那么也就不能去杀死它们。为了清除这样的进程，必须纠正底下的问题，或者重启系统。

僵进程是已经执行完毕但还没有让它们的状态被收集起来的进程。如果看到有僵进程挂在那里，那么就要用 **ps** 查看它们的 PPID，找出它们的来源。

处于停止状态的进程从管理上来说是被禁止运行的。进程一接收到 STOP 或 TSTP 信号，就进入停止状态，并可以使用 CONT 信号来重新启动处于停止状态的进程。处于停止状态与睡眠状态类似，但除了让另外某个进程来唤醒（或者终止）进程以外，它是不能够脱离停止状态的。

www.TopSage.com

4.6 nice 和 renice: 影响调度优先级

进程的“谦让度”是以数字形式给内核的暗示，通过它来表明一个进程在同其他进程竞争 CPU 时应该如何对待这个进程。它的这个奇怪名称源自于这样的事实：它确定了您准备对待系统上其他用户的谦让程度。高谦让值表示进程具有低优先级：您准备很友好地对待其他进程。低谦让值或者负值表示进程具有高优先级：对其他进程就不那么谦让了。谦让值的允许范围是-20～+19。

除非用户采取特殊动作，否则，新创建的进程就从它的父进程那里继承谦让值。进程的属主可以增加其谦让值，但不能够降低谦让值，哪怕是让进程返回到默认谦让值也不行。这种限制防止了具有低优先级的进程派生出高优先级的子进程。超级用户可以任意设置谦让值。

如今已经很少通过手工设置进程的优先级了。在 20 世纪 70、80 年代那些功能并不强大的系统上，影响性能的最重要因素就是在 CPU 上运行的是什么进程。现在，由于多数台式机具备了功能足够强大的 CPU，调度程序通常能够很好地完成服务所有进程的工作。在需要快速响应的情况下，开发人员可以使用新增的调度类型做更进一步的控制。I/O 性能还没有跟上不断加速的 CPU，所以现在大多数系统上的主要瓶颈已经变成了磁盘驱动器的速度问题了。遗憾的是，一个进程的谦让值并不会影响内核对其内存或 I/O 的管理，高谦让度的进程仍然比例不当地垄断了这些共享资源中的一部分。

进程的谦让值可以在创建进程时用 **nice** 命令来设置，并可以在执行时使用 **renice** 命令进行调整。**nice** 带一个命令行作为参数，而 **renice** 带 PID 或用户名作为参数。让人迷惑的是，**renice** 需要一个绝对优先级，而 **nice** 需要的则是优先级增量，然后把它加上 shell 的当前优先级，或者从 shell 的当前优先级中减去。

下面举几个例子：

```
$ nice -n 5 ~/bin/longtask      // 把优先级降低（提高谦让度）5
$ sudo renice -5 8829           // 把谦让值设为-5
$ sudo renice 5 -u boggs        // 把 boggs 的进程的谦让值设为 5
```

C shell 和另外一些常见的 shell 中（但不是 **bash**）内置了一种 **nice** 版本，使得情况变得复杂起来。如果不键入 **nice** 命令的完整路径，将得到 **nice** 的 shell 版本而不是操作系统的版本。这可能会令人困惑，因为 shell 的 **nice** 和操作系统的 **nice** 几乎总是使用不同的语法。shell 的 **nice** 要求它的优先级增量用*+incr* 或者*-incr* 来表达，而独立的 **nice** 命令则要求用**-n** 标志，后跟优先级增量[7]。

在现代世界中，最常见的使用 **nice** 的进程是 **xntpd**，它是时钟同步守护进程。由于 CPU 的及时性对它的任务非常重要，因此它通常运行在比默认值低 12 左右的谦让值上（也就是说，它的优先级比普通进程更高）。

如果某个进程“发狂”而使得系统的负荷平均值达到 65 的话，在能够运行命令去调查问题以前，可能需要使用 **nice** 来启动一个高优先级的 shell，否则，命令可能永远得不到运行的机会。

4.7 PS：监视进程

ps 是系统管理员监视进程的主要工具。用户可以用它显示进程的 PID、UID、优先级和控制终端。

[7] 实际上情况甚至更糟：独立的 **nice** 命令把 **nice -5** 解释成值为正 5 的增量，而 shell 的内置 **nice** 命令会把同一形式解释成值为负 5 的增量。

www.TopSage.com

大家网 TopSage.com

它还给出了有关一个进程正在使用多少内存、已经消耗了多少 CPU 时间以及它的当前状态（运行中、已停止、在睡眠等）的信息。在 **ps** 中，僵进程显示为<defunct>。

在各种 UNIX 变体上，**ps** 的表现显得千变万化，过去几年里，它的许多实现都变得相当复杂。为了努力适应已经习惯于其他系统上 **ps** 命令的用户，Linux 提供了一种混合型的 **ps** 版本，它能理解许多其他 **ps** 实现的选项集合，而且 Linux 能用一个环境变量告诉它假定采用哪一种风格。

不要被所有这些复杂性所吓住：复杂主要是对内核开发人员而言的，而不是对系统管理员而言的。即使您会频繁用到 **ps**，也只需知道几条特定的用法而已。

您可以用命令 **ps aux** 了解正在系统上运行的所有进程的全貌。下面举个例子（我们删掉了 START 列，而且只从输出中选了若干行）：

```
$ ps aux
  USER     PID  %CPU %MEM    VSZ   RSS   TTY      STAT  TIME  COMMAND
    root     1   0.1  0.2   3356   560   ?        S     0:00  init [5]
    root     2   0    0        0     0   ?        SN    0:00  [ksoftirqd/0]
    root     3   0    0        0     0   ?        S<    0:00  [events/0]
    root     4   0    0        0     0   ?        S<    0:00  [khelper]
    root     5   0    0        0     0   ?        S<    0:00  [kacpid]
    root    18   0    0        0     0   ?        S<    0:00  [kblockd/0]
    root    28   0    0        0     0   ?        S     0:00  [pdflush]
...
    root   196   0    0        0     0   ?        S     0:00  [kjournald]
    root  1050   0    0.1   2652   448   ?        S<s   0:00  udevd
    root  1472   0    0.3   3048  1008   ?        S<s   0:00  /sbin/dhclient -1
    root  1646   0    0.3   3012  1012   ?        S<s   0:00  /sbin/dhclient -1
    root  1733   0    0        0     0   ?        S     0:00  [kjournald]
    root  2124   0    0.3   3004  1008   ?        Ss    0:00  /sbin/dhclient -1
    root  2182   0    0.2   2264   596   ?        Ss    0:00  syslogd -m 0
    root  2186   0    0.1   2952   484   ?        Ss    0:00  klogd -x
     rpc  2207   0    0.2   2824   580   ?        Ss    0:00  portmap
 rpcuser  2227   0    0.2   2100   760   ?        Ss    0:00  rpc.statd
    root  2260   0    0.4   5668  1084   ?        Ss    0:00  rpc.idmapd
    root  2336   0    0.2   3268   556   ?        Ss    0:00  /usr/sbin/acpid
    root  2348   0    0.8   9100  2108   ?        Ss    0:00  cupsd
    root  2384   0    0.6   4080  1660   ?        Ss    0:00  /usr/sbin/sshd
    root  2399   0    0.3   2780   828   ?        Ss    0:00  xinetd -stayalive
    root  2419   0    1.1   7776  3004   ?        Ss    0:00  sendmail: accepi
...
```

中括号括起来的命令名不是真正的命令，而是按进程方式来调度运行的内核线程。表 4.3 解释了每个字段的意义。

另一组有用的选项是 **lax**，它提供了技术性更强的信息。而且它运行的速度也更快，因为它不必把每个 UID 都转换为用户名——如果系统已经因别的某个进程而停顿了，那么效率就显得很重要了。运行 **ps** 命令的开销一般都相当大。

表 4.3　　ps aux 命令输出的解释

字　段	内　容
USER	进程属主的用户名
PID	进程 ID
%CPU	该进程正在使用的 CPU 百分比
%MEM	该进程正在使用的实际内存百分比
VSZ	进程的虚拟大小

www.TopSage.com

续表

字 段	内 容
RSS	驻留集的大小（内存中页的数量）
TTY	控制终端的 ID
STAT	当前进程的状态： R=可运行 D=在等待磁盘（或者短期等待） S=在睡眠（<20s）T=被跟踪或者被停止 Z=僵进程 附加标志： W=进程被交换出去 <= 进程拥有比普通优先级更高的优先级 N=进程拥有比普通优先级更低的优先级 L=有些页面被锁在内存中 s=进程是会话的先导进程
START	启动进程的时间
TIME	进程已经消耗掉的 CPU 时间
COMMAND	命令的名称和参数[a]

a. 程序能够修改这项信息，因此它未必是实际命令行的确切表示。

下面给出了一个简短的例子，**ps lax** 的输出包括父进程 ID（PPID）、谦让值（NI）字段以及进程正在等待的资源（WCHAN）。

```
$ ps lax
  F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT   TIME COMMAND
  4     0     1     0  16   0   3356   560 select S      0:00 init [5]
  1     0     2     1  34  19      0     0 ksofti SN     0:00 [ksoftirqd/0
  1     0     3     1   5 -10      0     0 worker S<     0:00 [events/0]
  1     0     4     3   5 -10      0     0 worker S<     0:00 [khelper]
  5     0  2186     1  16   0   2952   484 syslog Ss     0:00 klogd -x
  5    32  2207     1  15   0   2824   580 -      Ss     0:00 portmap
  5    29  2227     1  18   0   2100   760 select Ss     0:00 rpc.statd
  1     0  2260     1  16   0   5668  1084 -      Ss     0:00 rpc.idmapd
  1     0  2336     1  21   0   3268   556 select Ss     0:00 acpid
  5     0  2384     1  17   0   4080  1660 select Ss     0:00 sshd
  1     0  2399     1  15   0   2780   828 select Ss     0:00 xinetd -sta
  5     0  2419     1  16   0   7776  3004 select Ss     0:00 sendmail: a
...
```

4.8 TOP：更好地监视进程

由于 **ps** 这样的命令只提供系统过去时间的一次性快照，因此，要获得系统上正在发生事情的“全景”往往是非常困难的。**top** 命令对活动进程以及其所使用的资源情况提供定期更新的汇总信息。例如：

```
top - 16:37:08 up  1:42,  2 users,  load average: 0.01, 0.02, 0.06
Tasks:  76 total,   1 running,  74 sleeping,   1 stopped,   0 zombie
Cpu(s):  1.1% us,  6.3% sy,  0.6% ni, 88.6% id,  2.1% wa,  0.1% hi,  1.3% si
Mem:   256044k total,  254980k used,    1064k free,   15944k buffers
Swap:  524280k total,        0k used,  524280k free,  153192k cached

  PID USER PR  NI  VIRT  RES  SHR  S %CPU %MEM TIME+  COMMAND
3175 root  15   0 35436  12m  4896 S  4.0    5.2  01:41.9 X
```

www.TopSage.com

```
 3421 root  25  10 29916 15m 9808 S  2.0  6.2 01:10.5 rhn-applet-gui
    1 root  16   0  3356 560  480 S  0.0  0.2 00:00.9 init
    2 root  34  19     0   0    0 S  0.0    0 00:00.0 ksoftirqd/0
    3 root   5 -10     0   0    0 S  0.0    0 00:00.7 events/0
    4 root   5 -10     0   0    0 S  0.0    0 00:00.0 khelper
    5 root  15 -10     0   0    0 S  0.0    0 00:00.0 kacpid
   18 root   5 -10     0   0    0 S  0.0    0 00:00.0 kblockd/0
   28 root  15   0     0   0    0 S  0.0    0 00:00.0 pdflush
   29 root  15   0     0   0    0 S  0.0    0 00:00.3 pdflush
   31 root  13 -10     0   0    0 S  0.0    0 00:00.0 aio/0
   19 root  15   0     0   0    0 S  0.0    0 00:00.0 khubd
   30 root  15   0     0   0    0 S  0.0    0 00:00.2 kswapd0
  187 root   6 -10     0   0    0 S    0    0 00:00.0 kmirrord/0
  196 root  15   0     0   0    0 S    0    0 00:01.3 kjournald
...
```

默认设置下，这些显示信息每 10s 更新一次。那些最活跃的进程显示在顶部。**top** 还接收键盘的输入并允许用户向进程发送信号和调整进程的谦让值，因此用户可以观察自己的操作是如何影响到机器的整体状态的。

root 用户能够以 **q** 选项去运行 **top**，以便把它提升到可能的最高优先级。当试图跟踪一个已经把系统拖垮的进程时，这种提升就非常有用。

4.9 /PROC 文件系统

Linux 版的 **ps** 和 **top** 命令都从**/proc** 目录读取进程的状态信息，内核把有关系统状态的各种有意义的信息都放在这个伪目录里。虽然这个目录叫做**/proc**（下面的文件系统类型也叫做“proc”），但是它里面的信息却并不局限于进程信息——内核产生的所有状态信息和统计数据都在这里。用户可以通过向/proc 下的适当文件写入数据的方法来修改某些参数——参见 28.4 节的例子。

虽然使用诸如 **vmstat** 和 **ps** 这样的前端命令是访问某些信息最方便的办法，但是有些不那么常用的信息就必须从/proc 目录下直接读取。应该花时间多浏览一下这个目录，以熟悉里面的各项内容。**man proc** 命令也能给出一些有用处的技巧和窍门。

由于内核随时都在创建**/proc** 下文件的内容（在读它们的同时），所以**/proc** 下的大多数文件在用 **ls -l** 命令列出时都显示为空。用户应该用 **cat** 或者 **more** 去看这些文件里的内容是什么。但是要注意——有几个文件包含或者链接到了二进制数据，如果直接去看的话，会搞乱终端模拟程序的显示。

进程特有的信息都分别被放到了按 PID 起名字的子目录里。例如，**/proc/1** 一定是包含 init 信息的目录。表 4.4 列出了各个进程最有用的文件。

表 4.4　　/proc 目录下的进程信息文件（数字编号的子目录）

文　件	内　容
cmd	进程正在执行的命令或者程序
cmdline[a]	进程的完整命令行（以 null 分隔）
cwd	链到进程当前目录的符号链接
environ	进程的环境变量（以 null 分隔）
exe	链到正被执行的文件的符号链接
fd	子目录，其中包含链到每个打开文件的描述符的链接
maps	内存映射信息（共享段、库等）

www.TopSage.com

续表

文 件	内 容
root	链到进程的根目录（由 **chroot** 设置）的符号链接
stat	进程的总体状态信息（**ps** 最擅长解析这些信息）
statm	内存使用情况的信息

a. 如果进程被交换出内存的话可能得不到。

在 **cmdline** 和 **environ** 文件里的各个部分用空字符（null）而不是换行符（newline）分隔。用户可以借助命令 **tr "\000" "\n"** 过滤这些文件的内容，使之可读性更好。

子目录 **fd** 里以符号链接形式表示进程打开的文件。连接到管道或者网络套接口的文件描述符没有相关联的文件名。内核提供一种通用的描述形式作为链接目标。

maps 文件用于确定一个程序链接到哪些库或者依赖于哪些库。

4.10 STRACE：追踪信号和系统调用

在传统的 UNIX 系统上，判断一个进程实际正在做什么相当困难。用户可能不得不根据从文件系统以及 **ps** 这样的工具取得的间接数据和经验来推测。与此相对照，Linux 能让用户通过 **strace** 命令直接观察一个进程，进程每调用一次系统调用，以及每接收到一个信号，这个命令都能显示出来。用户甚至可以把 **strace** 附在一个正在执行的进程上，监视一会儿该进程，再从进程上脱离，整个过程都不会影响那个进程[8]。

虽然系统调用出现在相当底层的地方，但是用户通常都可以从 **strace** 的输出里相当清楚地了解到进程的活动情况。例如，下面的记录是由 **strace** 附在一个活动的 **top** 进程上获得的：

```
$ sudo strace -p 5810
gettimeofday({1116193814, 213881}, {300, 0})            = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY) = 7
fstat64(7, {st_mode=S_IFDIR|0555, st_size=0, ...})      = 0
fcntl64(7, F_SETFD, FD_CLOEXEC)                         = 0
getdents64(7, /* 36 entries */, 1024)                   = 1016
getdents64(7, /* 39 entries */, 1024)                   = 1016
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)                          = 8
read(8, "1 (init) S 0 0 0 0 -1 4194560 73"..., 1023)    = 191
close(8)                                                = 0
...
```

strace 不仅能够显示进程每次调用的系统调用名，它还能解析参数，给出由内核返回的结果代码。

在本例中，**top** 先检查当前时间。然后打开**/proc** 目录，用 **fstat** 获得其信息，然后读取该目录的内容，由此获得当前正在运行的进程清单。**top** 接着用 **stat** 获得代表 **init** 进程的那个目录的信息，然后打开**/proc/1/stat** 读取 **init** 的状态信息。

4.11 失控进程

失控进程有两类：一类是过度占用了某种系统资源（例如 CPU 时间或磁盘空间等）的用户进程，

[8] 通常是这样。在有些情况下，strace 会中断系统调用。被监视的进程接下来必须准备重新发起调用。这是 UNIX 软件应遵守的标准规则，但不是总能看到这样的结果。

www.TopSage.com

另一类是突然发狂并呈现狂暴行为的系统进程。第一种失去控制的类型未必是一种故障，它可能仅仅是资源被过度占用了。系统进程总是被要求应该具有合理的行为方式。要了解有关失控进程的更多信息，请参见 25.4 节。

通过查看 **ps** 或者 **top** 的输出，能够确认是哪些进程占用了过多的 CPU 时间。如果明显发现某个用户进程正在消耗的 CPU 时间比预计的合理值多，那么请调查这个进程。在一台服务器或者一个共享系统上，第一步是联系该进程的属主，询问正在发生的事情。如果找不到属主，那么需要自己到周围去找找。尽管管理员通常应该避免查看用户的主目录，但在试图跟踪某个失控进程的源代码来查明它正在做什么的时候，这种做法是可以接受的。

之所以要在针对进程做出反应以前，先查明该进程正试图要做的事情，原因有两个。第一，该进程可能不但是合法的，而且对用户来说非常重要，这时候仅仅因为进程碰巧占用了大量的 CPU 时间而杀死它们并不合理。第二，进程可能是恶意的或具有破坏性的，如果是这样的话，管理员已经知道进程正在做的是什么事情（例如破译口令），因此就能够制止破坏行为。

如果不能够确定失控进程存在的原因，那么可以采用 STOP 信号把它挂起，并发送电子邮件给该进程的属主，解释所发生的事情。以后可以采用 CONT 信号来重新启动该进程。请注意，有些进程在长期睡眠以后可能被损坏，因此这种处理方法并不一定完全无风险。例如，某个进程醒来以后，可能会发现它的某些网络连接已经被断开。

如果某个进程使用 CPU 过量，但看起来又是在做合理的事情并工作正常，那么应该用 **renice** 命令给它更高的谦让值（也就是更低的优先级），并告诉该进程的属主以后再调整回该进程的谦让值。

相对于系统的物理存储器而言，过多占用内存的进程会导致严重的性能问题。用户可以使用 **top** 来检查进程占用内存的多少。**top** 输出的 VIRT 列给出了每个进程分配的虚拟内存量，RES 列给出了当前映射到特定内存页的那部分内存（驻留集）。

这两个数字都包括了像库这样的共享资源，从而让它们可能有误导性。在 DATA 列里能找到进程专门的内存消耗量更为直接的测定值。为了在 **top** 的输出里加上这一列的显示，要在运行 top 之后键入 **f**，从列表中选择 DATA。DATA 值表明了每个进程的数据和堆栈段占用的内存量，所以这个值对单个进程的针对性相当强（模块共享的内存段）。在观察随时间的增加情况的同时，也要观察绝对的内存量。

那些产生输出信息的失控进程能够填满整个文件系统，从而导致出现大量问题。当文件系统被填满时，控制台上将记录大量的消息，而试图向文件系统写入数据则会产生错误消息。

在这种情况下，要做的第一件事情是停止正在填满磁盘的那个进程。如果管理员在磁盘上一直保持着数量合理的空余空间作为回旋余地，那么当磁盘突然填满时，就能相当肯定地知道是出现了问题。没有类似于 **ps** 那样的命令可以告诉管理员哪个进程正在以最快的速度消耗掉磁盘空间，但有几种工具可以用来找到那些当前已打开的并且有进程正在使用它们的文件。更多的信息请参见后面对 **fuser** 和 **lsof** 的介绍。

作为管理员，可能想把所有看起来有问题的进程都挂起，一直到自己找到导致故障的那个进程为止。但当确定问题症结所在以后，要记得把其他没有问题的进程重新启动起来。一旦找到导致故障的那个进程，就要删除该进程创建的所有文件。

一个古老而又有名的“恶作剧”程序是从 shell 启动一个无限循环，做下面的事情：

```
while 1
    mkdir adir
    cd adir
    touch afile
end
```

有时，如果因为系统留下了已登录进入系统的终端，使得放开了对系统的访问，就可能会有人来

www.TopSage.com

运行这个循环，它并不消耗多少实际的磁盘空间，但它填满了文件系统的索引节点表（inode table），从而阻止其他用户创建新文件。对此，管理员除了清除错误结果并警告用户保护好自己的账号以外，并没有其他更多的措施可用。因为这个程序遗留下来的目录树通常非常大，用 **rm -r** 命令处理起来很费劲，所以可能需要编写一个脚本，先进入该目录树的底部，然后以回溯的方式删除各级目录。

如果问题出现在**/tmp** 中，并且您已经把**/tmp** 设置为一个单独的文件系统，那么可以采用 **mkfs** 命令来重新初始化**/tmp**，而不用试着删除所有的单个文件。有关文件系统管理的更多信息，请参见第 7 章。

4.12 推荐读物

BOVET, DANIEL P. AND MARCO CESATI. *Understanding the Linux Kernel (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2006.

4.13 习题

E4.1 解释一个文件的 UID 和一个正在运行的进程的真实 UID 以及有效 UID 之间的关系。除了文件的访问控制之外，进程的有效 UID 有什么用途？

E4.2 假定您所负责的站点中，有一个用户已经启动了一个长期运行的进程，它消耗了机器的很大一部分资源。

a）您如何意识到某个进程正在消耗资源？

b）假定异常的进程可能是合法的，不应该杀死它。给出您要把它“冻结”起来（在您调查期间，暂时停止它的运行）应该使用的命令。

c）随后，您发现这个进程属于您的老板，必须继续运行下去。给出您要继续执行这个任务应该使用的命令。

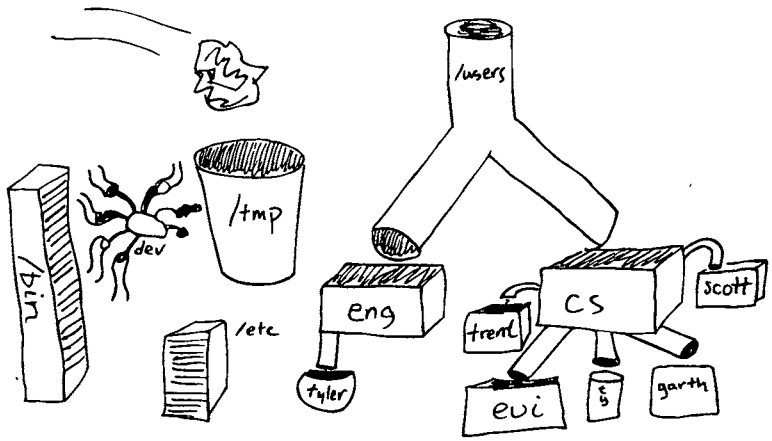
d）另一种可能是假定需要杀死这个进程。您会发送什么信号？为什么？如果您要保证这个进程确实已经被杀死了，该怎么做？

E4.3 找出一个能导致内存泄漏的进程（如果手头没有，就自己写一个）。用 **ps** 或者 **top** 来监视程序运行时的内存使用情况。

★E4.4 编写一个处理 **ps** 输出的 Perl 脚本，确定在系统上正在运行的进程总的 VSZ 和 RSS 值。这些数字和系统物理内存和交换空间的实际量有什么要的关系？

www.TopSage.com

第5章 文件系统



快速小测验：在“文件系统”中，您会找到下面的哪一项？

- 进程；
- 串行端口（串口）；
- 内核数据结构和调整参数；
- 进程间的通信通道。

如果系统是 Linux，那么答案是“以上全是”。当然，您也会在文件系统里找到一些文件[1]。

尽管文件系统的基本目的是表示和组织系统的存储资源，但程序员一直渴望在处理其他类型对象时，避免重新设计表示结构。事实已经多次证明，把那些对象映射到文件系统的名字空间里是自然和方便的做法。这种统一既有优点（一致的编程接口、易于从 shell 访问），也有一些缺点（有“自作自受”之嫌的文件系统实现）。但是，不管您是否喜欢，这已经成为了 UNIX（因此也是 Linux）的方式。

可以认为文件系统包括 4 种主要的组成部分：

- 名字空间——给事物取名，并按一种层次结构组织它们；
- API[2]——用来遍历和操作对象的一套系统调用；
- 安全模型——用来保护、隐藏和共享事物的方案；
- 实现——把逻辑模型同实际硬件联系到一起的软件。

[1] 或许更为准确的说法是这些东西都在文件系统内表示。在大多数情况下，文件系统作为一个结合点，把客户程序同它们所要用的驱动程序和服务程序联系起来。

[2] Application Programming Interface（应用程序编程接口），它是一个通用术语，用于描述由库、操作系统或软件包提供给程序员来调用的一组例程。

www.TopSage.com

Linux 定义了一个抽象的内核级接口，能够适应多种不同的后端文件系统。文件树的有些部分由传统的基于磁盘的实现来处理；其他部分则由内核中单独的驱动程序来负责。例如，NFS 文件系统由一个驱动程序来处理，这个驱动程序把被请求的操作转发到另外一台计算机的服务器。NFS（网络文件系统）将在第 16 章介绍。

遗憾的是，体系结构上的界线划分并不清晰，因此还有许多特殊的情形。例如，设备文件的存在给程序提供了一种同内核内部的驱动程序进行通信的方式。这些设备文件不是真正的数据文件，但它们由基本的文件系统驱动程序来处理，并且它们的特征都存储在磁盘上。如果现在根据过去几十年的经验重新实现文件系统的话，那么在细节方面会有些区别。

另外还有一个因素使问题变复杂（但最终还是有益的），即 Linux 支持不只一种基于磁盘的文件系统。现在最好的一类文件系统包括：作为大多数 Linux 发行版本默认文件系统的 ext3fs，还有 ReiserFS（参见 www.namesys.com 了解有关 ReiserFS 的更多信息）、IBM 的 JFS 和 SGI 的 XFS。ext3fs 之前的 ext2fs 比较老了，但所有的发行版本都支持它，将来很长一段时间里也仍然会保持对它的支持。

在 Linux 上也实现了许多外来的文件系统，比如在微软 Windows 上使用的文件系统 FAT 和 NTFS，以及 CD-ROM 上使用的 ISO-9660 文件系统。Linux 支持的文件系统数量超过了任何一种别的 UNIX 变体。它提供的丰富选择赋予了用户更多的灵活性，而且更易于和别的系统共享文件。

文件系统讨论起来是个内容很丰富的话题，我们要从几个不同的角度来介绍。本章告诉读者可以从系统上的什么地方找到什么样的文件，还告诉读者文件的特性、权限位的含义，以及一些查看和设置属性的基本命令的用法。在第 7 章里，读者会找到有关文件系统方面技术性更强的内容，比如磁盘分区。第 16 章将介绍 Linux 上常用的文件共享系统。读者还可以看看第 26 章，这一章讨论 Linux 为了和运行微软 Windows 的计算机共享文件所使用的软件。

虽然 Linux 上有那么多种不同文件系统的实现，可本章内容读起来就好像只有一种 Linux 文件系统一样，是不是有点奇怪呢。因为对于大多数现代的文件系统来说，要么是为了能以更快和更可靠的方式提供传统文件系统的功能，要么就是通过在标准文件系统的语义上加一层来增加更多的特性（有的文件系统两者都有），所以我们对诸多文件系统的实现不怎么了解。无论如何，现在太多的软件都依赖于本章介绍的模型，而不是我们没讲的模型。

5.1 路径名称

文件系统表现为单个统一的层次结构[3]：从目录/开始并通过若干数量的子目录继续向下扩展，/也叫做根目录。

为了找到某个特定文件而必须遍历的一系列目录，再加上文件名就形成了“路径名”。路径名既可以是绝对的（**/tmp/foo**），也可以是相对的（**book4/filesystem**）。相对路径认为是从当前目录开始。用户可能习惯于把当前目录看作是 shell 的特色，但实际上每个进程也都拥有一个当前目录。

术语文件（file）、文件名（filename）、路径名（pathname）、路径（path）或多或少都可以互换使用（或者至少在本书中可以互换使用）。文件名和路径可以用来指绝对路径和相对路径，路径名通常表示绝对路径。

文件系统的深度可以任意。不过，路径名每部分的长度一定不能超过 255 个字符，并且单个路径总长不能超过 4095 个字符。要访问超过这个长度的路径名，就必须 **cd** 到中间的目录并使用相对路径[4]。

[3] 这种单一层次结构的系统和 Windows 所采用的不一样，Windows 保留了特定于磁盘的名字空间概念。

[4] 这里再说明一下以防不清楚：多数文件系统的磁盘格式本身并没有对路径名称的总体长度进行限制。但是，访问文件系统的系统调用不允许它们的字符串参数的长度超过 4095 个字符。

www.TopSage.com

除了名称在长度上有限制和不能够包含“/”字符或空值以外，对文件和目录的命名基本上没有什么其他限制。特别是在一定程度上允许使用空格。由于 UNIX 长期以来有着以空白来分隔命令行参数的传统，所以当文件名中出现空格时，老软件可能会出现问题。

在文件名中有空白的情况过去主要出现在与 Mac 和 PC 共享的文件系统上，但是现在已经蔓延到 Linux 的文化之中，在一些标准的软件包里也能看到这样的文件名了。没别的办法，负责系统管理的脚本必须做好处理空白的准备。

在 shell 和在脚本中，只需用引号扩起带有空白的文件名，就能保持其完整性。例如，命令：

```
$ less "My excellent file.txt"
```

会把 **My excellent file.txt** 当作是 **less** 的单个参数。还可以用反斜线转移空格。常用 shell 所带的文件名补全功能（通常绑定到<tab>键上）就能替用户做转移。

在编写脚本的时候，要知道一个很有用的工具，这就是 **find** 命令的**-print0** 选项。这个选项和 **xargs -0** 一起使用就能让 **find/xargs** 命令组合不管文件名是否包含有空白都能争取执行。例如，下面的命令：

```
$ find /home -size +1M -print0 | xargs -0 ls -l
```

找到/home 分区下大小超过 1MB 的每个文件，按 ls 命令的长格式列出。

5.2 安装和卸载文件系统

文件系统由更小一些的块所组成，这些块儿也叫文件系统，每个这样的块都是由一个目录及其子目录和文件所组成的。正常情况下，根据上下文就可以明确判断出正在讨论的是哪种类型的“文件系统”，但为了清楚起见，我们将使用术语“文件树”来指文件系统的整个结构，而保留“文件系统”一词用来表示附加到文件树的块。

大多数文件系统是磁盘分区，但正如我们前面提到的那样，文件系统可以是遵循适当 API 的任何事物：网络文件服务器、内核组件、基于内存的磁盘模拟器等。Linux 甚至有一个不错的“loopback（环回）”文件系统，能让用户安装单个文件，就好像它们是独立的设备一样。

使用 **mount** 命令把文件系统附加到文件树上。**mount** 把现有文件树中的一个目录映射为新加入的文件系统的根，这个目录叫做安装点（mount point）。只要有另一个文件系统被安装在安装点上，那么就不能再访问到这个安装点以前的内容了。不过安装点通常是一些空目录。

例如，命令：

```
# mount /dev/hda4 /users
```

将把存储在磁盘分区上由**/dev/hda4** 表示的文件系统安装到路径**/users** 下。随后可以使用 **ls /users** 来查看这个文件系统的内容。

根据惯例，安装在某个特定系统上的文件系统清单保存在**/etc/fstab** 文件中。包含在这个文件中的信息让这些文件系统在引导时先检查（**fsck -A**）再自动安装（**mount -a**）到系统中。它还充当了磁盘上文件系统布局的文档资料这一角色，并可以使用像 **mount /usr** 这样的简写命令（文件系统要安装的位置可以在 **fstab** 中查到）。有关 **fstab** 文件的完整讨论，请参见 7.4 节。

umount 命令用来卸载文件系统。在大多数系统上，不能卸载正处于“busy（繁忙）”状态的文件系统。在该文件系统中不能有任何打开的文件，也不能有任何进程的当前目录，并且如果文件系统包含有可执行的程序，那么这些程序也不能够处于运行状态。

2.4.11 版以后的 Linux 内核定义了一种“lazy（缓慢）”的卸载方式（用 **umount -l** *命令调用），在*

www.TopSage.com

从目录名的层次结构中删除某个文件系统的时候，先要让当前所有对该文件系统内文件的访问都关闭，才能真正卸载这个文件系统。首先，不能保证当前的访问都会自行关闭。其次，“半卸载”状态也给使用文件系统的程序带来了不一致的文件系统语义。它们可以通过已有的文件句柄执行读写操作，却不能打开新文件或者执行其他文件系统操作。

如果内核抱怨说用户正在试图卸载处于繁忙状态的文件系统，可以运行 **fuser** 命令来查明原因。当以**-mv** 标志加一个安装点为参数来调用 **fuser** 命令时，它会显示出正在使用该文件系统上的文件或目录的每个进程的 PID：

```
$ fuser -mv /usr
        USER  PID  ACCESS COMMAND
/usr    root  444  ....m  atd
        root  499  ....m  sshd
        root  520  ....m  lpd
        . . .
```

ACCESS 列的字母代码表示每个进程对 **unmount** 命令产生的反应。表 5.1 介绍了每种代码的含义。

表 5.1 **fuser 给出的反应代码**

代 码	含 义
f	进程有一个为了读或者写而打开的文件
c	进程的当前目录在这个文件系统上
e	进程目前在执行一个文件
r	进程的根目录（用 **chroot** 命令设置）在这个文件系统上
m	进程已经映射了一个文件或者共享库（通常是一个非主动的可执行文件）

为了准确地判断带来麻烦的进程是什么，只要用 **fuser** 返回的 PID 清单运行一下 **ps** 就行了。例如：

```
$ ps -fp "444 499 520"
    UID  PID  PPID  C  STIME  TTY  TIME      CMD
daemon  444  1     0  Apr11  ?    00:00:00  /usr/sbin/atd
  root  499  1     0  Apr11  ?    00:00:23  /usr/sbin/sshd
    lp  520  1     0  Apr11  ?    00:00:00  [lpd]
```

引号强迫 shell 将 PID 清单作为一个参数传递给 **ps**。

fuser 也能报告特定文件（与整个文件系统相反）的使用，语法是 **fuser -v** *filename*。**fuser** 也接受 **-k** 选项，杀死每个带来麻烦的进程（或者向它们发送一个信号）。危险——您必须是 root（或者使用 **sudo**，参见 3.4.2 节）。

fuser 的替代工具是程序 **lsof**，它是由普渡大学（Purdue University）的 Vic Abell 编写的。**lsof** 在许多不同的 UNIX 和 Linux 变体上都能运行，这使得它很适合由必须在多种系统上运行的脚本调用。**lsof** 是一种比 **fuser** 更复杂和先进的程序，它的输出相应地也详细一些。

脚本在搜索特殊信息的时候也可以直接去读**/proc** 下的文件。不过，更简单、移植性更好的一种做法是使用 **lsof –F** 命令让 **lsof** 的输出有格式，从而易于分析。再加上其他命令行标志来限定只请求所需的信息。

5.3 文件树的组织

UNIX 家族的文件系统还从来没有被很好地组织过。各种互不兼容的命名约定在同时使用，整个名字空间中随意散落着不同类型的文件。在许多情况下，文件是按照其功能而不是按照被修改的可能

www.TopSage.com

性进行划分的，这使得操作系统的升级变得困难起来。例如，**/etc** 目录中既包含了一些从来就不需要定制的文件，又包含了一些完全是本地性的文件。如何知道哪些文件在升级过程中应该保存呢？是的，管理员确实必须知道……

像**/var** 这样的新增目录已经帮助解决了一些问题，但是大多数系统仍然处于缺乏组织的状态。尽管如此，每一个文件还都有其从文化意义上看是合适的位置。在 Linux 下，不要弄乱文件树的默认结构尤其重要，因为软件包和它们的安装工具经常对文件的位置做出大胆的假设（就像其他系统管理员那样！）。

根文件系统包括根目录和最小的一组文件和子目录，包含内核的那个文件位于根文件系统中的**/boot** 目录下，它的名字通常以 **vmlinuz** 开头[5]。存放设备文件的**/dev** 目录（除了**/dev/pts**，它单独安装）、存放关键系统文件的**/etc** 目录、存放重要工具的**/sbin** 和**/bin** 目录，有时候还有用来存放临时文件的**/tmp** 目录都是根文件系统的一部分。有关配置内核的更多信息请参见第 28 章。

目录**/usr** 和**/var** 也非常重要。**/usr** 是存放大多数标准程序的地方，它还有其他一些内容，例如在线用户手册和绝大多数数库文件。把**/usr** 作为单独的文件系统并非严格要求，但出于管理上的方便，几乎总是这么设置。为了让系统能够最终启动到多用户模式，必须有**/usr** 和**/var** 这两个目录。参考 7.4 节了解为什么最好进行分区，以及进行分区的一些指导。

/var 存放有假脱机目录、日志文件、记账信息和其他各种快速增长或变化的东西以及随主机不同而有所不同的东西。由于**/var** 包含有日志文件，而日志文件会随着故障出现的次数增多而增长，所以，如果可行的话，把**/var** 放置在它自己的文件系统上是一个好主意。

用户的主目录应该保存在单独的文件系统上，这个单独的文件系统通常安装在根目录中。还可以用单独的文件系统存储占用空间较大的东西，例如源代码库和数据库等。

表 5.2 中列出了一些比较重要的标准目录。

处于不断发展中的文件系统层次结构标准（Filesystem Hierarchy Standard）正在努力编制标准目录，使之合理，并加以说明。在要确定该把什么东西放在哪儿的时候，最好参考一下这个资源。我们在 11.6 节开始讨论设计本地文件系统结构时还要考虑的其他一些问题。

表 5.2　　标准目录及其内容

路 径 名	内　容
/bin	获得最小的系统可操作性所需要的命令
/boot	内核和加载内核所需的文件
/dev	终端、磁盘、调制解调器等的设备项
/etc	关键的启动文件和配置文件
/home	用户的主目录
/lib	C 编译器的库和部分 C 编译器
/media	可移动介质上文件系统的安装点
/opt	可选的应用软件包（尚未广泛采用）
/proc	所有正在运行进程的映像
/root	超级用户的主目录（经常就是 / ）
/sbin	引导、修复或者恢复系统的命令
/tmp	每次重新引导就消失的临时文件
/usr	次要文件和命令的层次结构

[5] 以前/boot 一度常作为一个独立的文件系统，这样做大多是因为内核必须放在靠近启动盘的开头处，好让 BIOS 能访问到。现代的 PC 不再有这个问题了，/boot 更常作为根文件系统的一部分。

www.TopSage.com

续表

路 径 名	内 容
/usr/bin	大多数命令和可执行文件
/usr/include	编译C程序的头文件
/usr/lib	库，供标准程序使用的支持文件
/usr/local	本地软件（用户所编写或者安装的软件）
/usr/local/bin	本地的可执行文件
/usr/local/etc	本地系统配置文件和命令
/usr/local/lib	本地的支持文件
/usr/local/sbin	静态链接的本地系统维护命令
/usr/local/src	**/usr/local/*** 的源代码
/usr/man	联机用户手册
/usr/sbin	不太关键的系统管理命令和修复命令
/usr/share	多种系统共同的东西（只读）
/usr/share/man	联机用户手册
/usr/src	非本地软件包的源代码
/var	系统专用数据和配置文件
/var/adm	各种不同的东西：日志、系统设置记录、奇怪的管理信息
/var/log	各种系统日志文件
/var/spool	供打印机、邮件等使用的假脱机目录
/var/tmp	更多的临时空间（在重新引导以后，文件予以保留）

5.4 文件类型

Linux定义了7种文件类型。即使在开发人员把精彩的新内容添加到文件树中（比如在**/proc**下列出的进程信息）的时候，仍然必须让新内容看起来像下面7种类型之一：

- 普通文件；
- 目录；
- 字符设备文件；
- 块设备文件；
- 本地域套接口；
- 有名管道（FIFO）；
- 符号链接。

用户可以用命令 **ls -ld** 来判断现有文件的类型。**ls** 命令输出的第一个字符表示类型。下面的例子表明**/usr/include** 是一个目录：

```
$ ls -ld /usr/include
drwxr-xr-x   27 root   root     4096 Jul 15 20:57   /usr/include
```

ls 使用表5.3中给出的代码来表示文件的各种类型。

www.TopSage.com

表 5.3 ls 命令使用的文件类型代码

文件类型	符号	创建方式	删除方式
普通文件	-	编辑器，cp 等	rm
目录	d	mkdir	rmdir, rm -r
字符设备文件	c	mknod	rm
块设备文件	b	mknod	rm
本地域套接口	s	socket(2)	rm
有名管道	p	mknod	rm
符号链接	l	ln -s	rm

由表 5.3 可见，**rm** 是删除不要文件的通用工具。不过，怎样删除一个名为**-f** 的文件呢？它在大多数文件系统中都是一个完全合法的文件名，但是 **rm -f** 却不能起作用，因为**-f** 被解释成了一个 **rm** 的标志。答案是，要么用更完整的路径名（比如**./-f**）来指出文件，要么用 **rm** 的**--**参数来告诉它后面的所有东西都是文件名而不是选项（例如，**rm -- -f**）。

包含控制字符的文件名带来了一个类似的问题，因为从键盘复制这些名字很困难乃至不可能。在这种情况下，可以使用 shell 的通配方式（模式匹配）来确定要删除的文件。在采用模式匹配的时候，使用 **rm** 的**-i** 选项，让 **rm** 确认每个文件的删除操作是一个好主意。这项功能可以防止删除偶然匹配模式但又是“好”的任何文件。例如，要删除一个名为 **foo<Control-D>bar** 的文件，可以使用：

```
$ ls
foo?bar      foose      kde-root

$ rm -i foo*
rm: remove `foo\004bar'? y
rm: remove `foose'? n
```

注意，**ls** 将控制字符显示为一个问号，这会有点儿欺骗性[6]。如果忘记了“**?**”是一个 shell 要进行模式匹配的字符，而试着用 **rm foo?bar**，那么可能会删除不只一个文件（虽然本例中不是这样）。**-i** 应该是常用的选项！

为了删除名字最难对付的文件，可能需要求助 **rm -i ***这条命令。

还有一种方法能删除名字古怪的文件，就是通过访问文件系统的其他接口，比如 **emacs** 的 dired 模式，或者像 Nautilus 这样的可视化工具。

5.4.1 普通文件

普通文件只是一个装字节的包而已；Linux 并没有就其内容规定任何结构。文本文件、数据文件、可执行程序和共享库都作为普通文件存储。普通文件既能顺序存取，也能随机存取。

5.4.2 目录

目录包含按名字对其他文件的引用。用户可以使用 **mkdir** 命令来创建目录，使用 **rmdir** 命令来删

[6] **ls -b** 命令以八进制数显示特殊字符，如果需要特意辨别它们的话，这个命令就可以帮上忙。<Control-A>是 1（八进制为 \001）、<Control-B>是 2，以此类推。

www.TopSage.com

除空目录，使用 **rm -r** 命令来删除非空目录。

特殊项“.”和“..”分别代表目录本身和它的父目录，它们不可以移动。由于根目录没有父目录，所以“/..”和“/.”是等价的（都等同于/）。

文件的名称实际上存储在它的父目录中，而不是和文件本身存储在一起。事实上，在同一时间，不只一个目录（或者是单个目录中的不只一个目录项）能够引用一个文件，并且引用可以拥有不同的名称。这样的安排产生了这样的错觉：一个文件同时存在于多个位置。

这些多出来的引用（链接）跟原来的文件是没有什么区别的；在 Linux 看来，它们都是等同的。Linux 维护着指向每个文件的链接的计数，在该文件的最后一个链接被删除之前不释放该文件的数据块。Linux 的链接不能够跨过文件系统的边界。

现在，这种类型的引用通常叫做“硬链接”，以此把它们同符号链接区分开来，符号链接将在下面介绍。我们采用 **ln** 命令来创建硬链接，采用 **rm** 命令来删除硬链接。

如果记住 **ln** 是 **cp** 的镜像，那么就很容易记住 **ln** 的语法。命令 **cp oldfile newfile** 创建 **oldfile** 的一个副本，该副本叫做 **newfile**。**ln oldfile newfile** 让名称 **newfile** 成为对 **oldfile** 新增的一个引用。

要理解硬链接并不是文件的一种独特类型，这一点很重要。文件系统只是允许不只一个目录项指向某个特定的文件，而不是定义了一个单独的称为硬链接的“事物”。除了文件的内容之外，所有链接之间还共享这个文件的根本属性，比如归属关系和访问权限。

5.4.3 字符设备文件和块设备文件

设备文件让程序能够同系统的硬件和外围设备进行通信。在配置内核的时候，那些知道怎样同系统的每个设备进行通信的模块就被链入内核[7]。用于某个特定设备的模块叫做设备驱动程序，它负责管理该设备的凌乱细节。有关设备及其驱动程序的更多信息，请参见第 28 章。

设备驱动程序提供了一个标准的通信接口，该接口看起来就好像是一个普通文件。当内核接到一个对字符或块设备文件的请求时，它就简单地把这个请求传递给适当的设备驱动程序。不过，区分设备文件和设备驱动程序是很重要的。设备文件只是用来同设备驱动程序进行通信的结合点。它们并不是设备驱动程序本身。

字符设备文件让与之相关的驱动程序做它们自己的输入和输出缓冲。块设备文件由处理块数据 I/O 的驱动程序使用，并要求内核为它们提供缓冲。以前，有些类型的硬件既可以表示为块设备文件，也可以表示为字符设备文件。但是现在很少有这样的配置了。

设备文件用两个数字来表示其特征，这两个数字分别叫做主设备号和次设备号。主设备号告诉内核该文件访问哪个驱动程序，次设备号告诉驱动程序对哪个物理单元寻址。例如，Linux 系统上的主设备号 6 表示并行端口驱动程序。第一个并行端口（**/dev/lp0**）的主设备号为 6，次设备号为 0。

有些设备驱动程序以它们高兴的方式来解释传给它们的次设备号。例如，磁带驱动程序经常使用次设备号来确定关闭该设备文件时是否应该回卷磁带。

设备文件可以使用 **mknod** 命令来创建，使用 **rm** 来删除。不过基本上没有必要手工创建设备文件。大多数发行版本使用 **udev** 根据内核对硬件的检测结果自动创建和删除设备文件。通过限制/dev 目录下虚设的设备文件数量，以及确保分配给文件的设备号符合内核期望的做法，**udev** 就能保持/dev 目录的整洁。参见第 28 章了解更多这方面的知识。

万一需要手工创建设备文件时，有个叫做**/dev/MAKEDEV** 的老脚本作为 udev 的备用工具挺不错的。这个脚本能为各类设备定出符合规范的设备文件名和设备号，因此用户不必亲自去检查结果。例

[7] 这些模块也可以被内核动态地加载。

www.TopSage.com

如，**MAKEDEV pty** 这条命令能给伪终端创建设备文件。

如果需要得知一个驱动程序的主次设备号，在该驱动程序的手册页中第 4 节（例如，**man 4 tty**）里就能找到这方面的信息。

5.4.4 本地域套接口

套接口（socket）就是在进程之间让它们以“干净卫生”的方式进行通信的连接。Linux 提供了几种不同类型的套接口，其中大多数涉及使用网络。本地域套接口只能从本地主机访问，并且是通过文件系统对象而不是网络端口来使用。它们有时称为“UNIX 域套接口（UNIX domain socket）”。

尽管套接口文件对于其他进程是可见的目录项，但是通信连接之外的进程不能够读写它们。使用本地域套接口的一些标准工具有打印系统、X Windows 系统和 syslog。有关系统日志的更多信息，请参见第 10 章。

本地域套接口由系统调用 **socket** 创建，当套接口不再有任何用户时，可以使用 **rm** 命令或系统调用 **unlink** 来删除它。

5.4.5 有名管道

与本地域套接口类似，有名管道能让运行在同一主机上的两个进程之间进行通信。它们也称为“FIFO 文件”（FIFO 是“first in，first out[先入先出]”的缩写）。有名管道使用 **mknod** 来创建，使用 **rm** 命令来删除。

5.4.6 符号链接

符号链接或者叫做“软”链接通过名字指向文件。当内核在查找路径名的过程中遇到符号链接时，它就把它的注意力重定向到作为该链接的内容而存储的路径名上。硬链接和符号链接之间的区别在于：硬链接是直接引用，而符号链接是通过名称进行引用，符号链接跟它们指向的文件是不同的。

符号链接使用 **ln -s** 来创建，使用 **rm** 来删除。由于符号链接能够包含任意的路径，所以它们可以指向其他文件系统上的文件或者指向不存在的文件。几个符号链接还可以形成一个环。

符号链接既可以包含绝对路径，也可以包含相对路径，例如，

```
# ln -s archived/secure /var/log/secure
```

这个命令采用相对路径把**/var/log/secure** 链接到**/var/achived/secure**。它用一个目标路径“**archived/secure**”创建了一个符号链接**/var/log/secure**，结果见下面这条 **ls** 命令的输出：

```
$ ls -l /var/log/secure
lrwxrwxrwx 1 root root 18 2005-07-05 12:54 /var/log/secure -> archived/secure[8]
```

整个**/var/log** 目录可以移动到另外某个位置而不会让这个符号链接不起作用（建议不要移动那个目录）。

认为 **ln -s** 的第一个参数与自己的当前工作目录有关是一个常见错误。**ln** 不把它解析为一个文件名，它只是被逐字用作符号链接的目标。

[8] ls 显示的符号链接的文件权限为 lrwxrwxrwx，这个权限是虚设的。创建、删除链接或者跟随链接的权限都由包含链接的目录控制，而读写和执行链接目标的权限由目标自己的权限决定。因此，符号链接不需要（也没有）它自己的任何权限信息。

www.TopSage.com

TopSage.com

5.5 文件属性

在传统的 UNIX 和 Linux 文件系统模型中，每个文件都有一组 9 个权限位用来控制谁能够读写和执行该文件的内容。这 9 位和另外影响到可执行程序运行的 3 个权限位一起，构成了文件的“模式”（mode）。

这 12 个模式位和 4 位的文件类型信息一起保存在一个 16 位的字中。这 4 个文件类型位在文件创建时设定并且不能修改，但是 12 个模式位可以由文件的属主或超级用户使用 **chmod**（改变模式）命令来修改。使用 **ls -l**（对于目录来说，使用 **ls -ld**）可以查看这些位的值。5.5.4 节中给出了一个例子。

5.5.1 权限位

9 个权限位用来确定可以由谁对文件执行什么样的操作。（尽管 Linux 现在在所有主要的文件系统上都支持访问控制列表，参见 5.6 节），传统的 UNIX 不能逐个用户地去设置权限。而是为文件的属主、文件的属组和其他每个人设置访问权限集合。每个集合有 3 位：读取位、写入位和执行位。

采用八进制数字来讨论文件的访问权限很方便，因为一个八进制数字的每一位代表 3 位，而每组权限位中正好有 3 位。最前面的 3 位（对应的八进制值为 400、200 和 100）控制属主的访问权限。第二个三位组（40、20 和 10）控制组的访问权限。最后面的三位组（4、2 和 1）控制其他每个人的访问权限。在每个三位组中，高位是读取位，中间位是写入位，低位是执行位。

每个用户只能够划归为这 3 个三位组中的一组，使用最具体的权限。例如，一个文件的属主（owner）拥有的访问权限由属主权限位而不是组权限位所确定。“其他”（other）和“组”（group）类别有可能拥有比属主更多的访问权限，但这样的配置很少使用。

在普通文件上，读取位允许打开该文件并读取它的内容。写入位允许修改或删截该文件的内容。不过，能否删除和重命名（或者删除后再重建）该文件则由该文件父目录上的权限设置所控制（因为名字到数据空间的映射关系实际上存储在父目录中）。

执行位允许执行文件。可执行文件有两种类型：一种是二进制的，CPU 能够直接运行它；另外一种是脚本，脚本必须由 shell 或其他某种程序来解释。按照惯例，脚本以下面形式的一行代码开始：

```
#!/usr/bin/perl
```

这行代码指定了一个合适的解释程序。没有指定解释程序的那些非二进制可执行文件（由用户的 shell）假定为 **bash** 或者 **sh** 脚本文件[9]。

对于目录来说，当分析路径名时，执行位（在这种上下文环境中，也经常叫做“搜索[search]”位或“扫描[scan]”位）的作用是控制是否能够进入或通过该目录，而不是控制能否列出它的内容。读取位和执行位的组合的作用才是控制是否列出目录中的内容。写入位和执行位的组合则允许在目录中创建、删除和重新命名文件。

5.5.2 setuid 和 setgid 位

八进制值为 4000 和 2000 的两位是 setuid 位和 setgid 位。如果在可执行文件上设置这两位，那么

[9] 内核能够理解#!语法，碰到它后会直接开始操作。不过，如果没有完全和正确地指定解释程序，那么内核将拒绝执行文件。接下来 shell 会通过调用 **sh** 来再次尝试执行这个脚本。

www.TopSage.com

它们能让程序访问运行它们的用户本来无权访问的文件和进程。可执行文件的 setuid/setgid 机制在 3.1 节介绍。

在某个目录上设置了 setgid 位以后，在这个目录中新创建的文件具有该目录的属组权限而不是创建该文件的用户的默认属组。这项约定使得在几个用户之间（只要这些用户都属于一个共同的组）共享一个目录中的文件变得更加简单。对 setgid 位的这种解释跟它在可执行文件上设置时的含义没有什么关系，但不要混淆了这两种意义。

您还可以在非可执行的纯文本文件上设置 setgid 位，在该文件被打开时请求特殊的锁定操作。但是，我们从未见到这项功能被用过。

5.5.3 粘附位

八进制值为 1000 的位叫做“粘附位”。在早期的 UNIX 系统上，它作为限定符对于可执行文件来说很重要。不过，粘附位的含义现在已经过时了，现代操作系统已经悄然地忽略了它。

如果在目录上设置了这个粘附位，那么除非您是该目录的属主、该文件的属主或者是超级用户，否则文件系统不会允许删除或重新命名该目录中的文件。在这个目录上拥有写入权限是不够的。这项约定有助于让像**/tmp** 之类的目录变得多少有些隐私性和安全性。

5.5.4 查看文件属性

文件系统为每个文件维护大约 40 项单独的信息，但其中的大多数只是对于文件系统本身有用。作为系统管理员，主要关心的是链接数、属主、属组、模式、大小、最后访问时间、最后修改时间和类型。所有信息可以使用 **ls -l**（或者对于目录来说是 **ls -ld**）来查看。

文件系统还为每个文件维护了属性改变的时间。给这个时间起的习惯名称（ctime，代表 change time）使得一些人以为它是文件的创建时间。遗憾的是，它不是，它只是记录文件属性（属主、模式等）最后改变的时间（注意跟修改文件内容的时间区分开来）。

考虑下面的例子：

```
$ ls -l /bin/gzip
-rwxr-xr-x      3 root    root     57136 Jun 15  2004 /bin/gzip
```

第一个字段指定该文件的类型和模式。第一个字符是短划线，因此这个文件是普通文件。参考表 5.3 了解其他文件类型的代码。

这个字段中接下来的 9 个字符就是 3 组权限位。这些组的顺序是“属主-属组-其他”，每组中位的次序是“读-写-执行”。尽管这些位只有二进制的值，但 **ls** 用字母来显示它们：字母 r 表示读取、字母 w 表示写入、字母 x 表示执行。在这个例子中，属主对该文件拥有全部的访问权限，其他每个人只拥有读取和执行的权限。

如果已经设置了 setuid 位，那么，表示属主的执行权限的 x 将用一个 s 来替代。如果 setgid 位已经被设置的话，那么表示组执行权限的 x 也会被 s 所替代。如果该文件的粘附位已经被打开，那么权限的最后那个字符（就是规定“其他人”执行权限的字符）显示为 t。如果设置了 setuid、setgid 或粘附位中的一个，但又没有设置其相应的执行位，那么这些位就显示为 S 或 T。

列表中的下一个字段是该文件的链接数目。在示例中是 3，表示**/bin/ gzip** 是这个文件 3 个名字中的一个（另外两个名字是**/bin/gunzip** 和**/bin/zcat**）。每次建立一个该文件的硬链接时，链接数目就增加 1。

所有目录将至少拥有两个硬链接：来自父目录的链接和来自目录本身内部的特殊文件“.”的链接。符号链接不影响链接数目。

www.TopSage.com

在 **ls** 输出中接下来的两个字段是文件的属主和属组。在这个例子中，该文件的属主是 root，该文件又属于组 root。文件系统实际上保存的是用户 ID 和组 ID 的编号，而不是字符串名。如果不能够确定数字 ID 号对应的文字（名字），那么这些字段将包含数字 ID 号。如果拥有该文件的用户或组已经从**/etc/passwd** 或**/etc/group** 文件中删除的话，那么可能会出现这种情况。出现这种情况还可能预示着 NIS 或者 LDAP 数据库（如果用到这个数据库的话）有问题，具体请参见第 17 章。

接下来的字段是文件以字节为单位的大小。示例中的文件长度是 57 136 字节，大约 56K[10]。接下来的字节是文件最后被修改的日期：2004 年 6 月 15 日。列表中最后那个字段是该文件的名称**/bin/gzip**。

如果是设备文件，**ls** 的输出稍微有些不同，例如：

```
$ ls -l /dev/tty0
crw-rw----   1 root     root     4,  0 Jun 11   20:41 /dev/tty0
```

大多数字段是一样的，但不显示文件的大小，而是显示其主设备号和次设备号。**/dev/tty0** 是设备驱动程序 4（在这个系统上，是终端驱动程序）所控制的第一个虚拟控制台。

ls 有一个选项**-i**，它用于查看硬链接，这个选项让 **ls** 显示每个文件的“索引节点号”。我们没有必要过多地深究有关文件系统实现的具体细节，我们只是说索引节点号是列出文件系统里所有文件的一张表中的一个索引。索引节点就是目录项所指向的东西，同一文件的硬链接项具有相同的索引节点号。要勾勒出链接复杂的网状关系，需要使用 **ls -li** 命令来给出链接数目和索引节点号，和 **find** 一起来找到匹配结果[11]。

系统自动跟踪修改时间戳、链接数目和文件大小信息的变化。相反地，权限位、归属关系和属组权只有当它们分别采用 **chmod**、**chown** 和 **chgrp** 命令修改时才改变。

其他一些应该知道的 **ls** 重要选项有：**-a** 列出一个目录的所有项（包括名字以点开头的文件），**-t** 按照时间对文件排序（或者**-tr**，按照时间逆序排列），**-F** 以区分目录和可执行文件的方式显示文件名，**-R** 逆序显示，以及**-h** 以方便人们阅读的形式显示文件大小（例如，8K 或者 53M）。

5.5.5 chmod：改变权限

chmod 命令改变文件的权限。只有文件的属主和超级用户才能够修改它的权限。为了在早期的 UNIX 系统上使用该命令，就不得不学一点八进制记法的知识，但该命令的当前版本既能够接受八进制的记法，也可以接受易于记忆的语法。一般说来，八进制的语法对于系统管理员来说更加方便，但它只可以用来指定权限位的绝对值。易于记忆的语法既可以修改一些位而又不影响其他位。

chmod 的第一个参数是要给文件分配的访问权限的说明，第二个参数及其以后的参数是要改变访问权限的文件的名字。在采用八进制记法的情况下，说明的第一个八进制位代表属主的权限，第二个八进制位代表组的权限，第三个八进制位代表其他每个人的权限。如果想要打开 setuid、setgid 或粘附位，则要使用 4 个而不是 3 个八进制位，由 3 个特殊位来构成第一个八进制位。

表 5.4 说明了每个三位组的 8 种可能组合，其中的 r、w 和 x 分别代表读取、写入和执行。

[10] K 表示千，它是 1000 的公制度量前缀。然而，计算机世界把它误用成了 2^{10}，即 1 024。与此类似，计算机的 1 兆字节实际上不是 1 百万字节，而是 2^{20}，即 1 048 576 个字节。国际电子技术委员会正在推行一组明确地基于 2 的幂的新数字前缀（例如 kibi 和 mebi）。现在看来似乎过去的习惯用法不大可能会改变。让情况更为混乱的是，甚至以 2 的幂为单位也不一定得到统一使用。RAM（随机存取存储器）是以 2 的幂为单位来表示的，但网络带宽却始终是用 10 的幂为单位来表示。制造商用 10 的幂为单位来报存储空间的大小，但是别人却会按 2 的幂来计算（译者注：这也是硬盘格式化后的空间要比标称值小一些的原因）。

[11] 尝试使用 **find** *mountpoint* **-xdev -inum** *inode* **-print**。

www.TopSage.com

表 5.4 chmod 的权限代码

八进制	二进制	权限	八进制	二进制	权限
0	000	---	4	100	r--
1	001	--x	5	101	r-x
2	010	-w-	6	110	rw-
3	011	-wx	7	111	rwx

例如，**chmod 711 myprog** 赋予属主所有的权限，而只给其他每个人赋予执行权限[12]。

chmod 的助记语法的全部信息可以在 **chmod** 的手册页中找到。其规则的一些示例如表 5.5 所示。

表 5.5 chmod 助记语法的例子

规则	含义
u+w	为文件的属主添加写入的权限
ug=rw,o=r	赋予属主和属组读取/写入的权限，赋予其他人读取的权限
a-x	删除全部 3 种类别用户（属主/属组/其他人）的执行权限
ug=srx,o=	设置文件的 setuid 和 setgid 位，并且只给属主和属组赋予读取/执行的权限
g=u	让属组的权限跟属主的权限完全一样

使用助记语法的困难之处在于要记住 **o** 代表“owner（属主）”还是“other（其他人）”（正确答案是其他人）。只要类比 UID 和 GID 记住 **u** 和 **g**，那么就只剩下一种可能了。

您还可以通过类比一个现有的文件来指定要分配的权限模式。例如，**chmod --reference=filea fileb** 让 **fileb** 的权限模式和 **filea** 的权限模式一样。

chmod 可以使用**-R** 选项递归地更新某个目录下文件的权限。不过，这可比它看上去更需慎重对待，因为包含进来的文件和目录可能不是全都有相同的属性（例如，有些文件可能是可执行文件，而别的文件可能是文本文件）。采用**-R** 选项时使用助记语法格外有用，因为没有明确设置的任何位，其值都保持不变。例如：

```
$ chmod -R g+w mydir
```

给 **mydir** 及其所有内容增加了属组的写入权限，同时又不会弄错目录以及程序的执行位。

5.5.6 chown：改变归属关系和组

chown 命令改变文件和文件的属组所有权。**chown** 的语法跟 **chmod** 类似，只不过它的第一个参数以 *user:group* 的形式指定了新的属主和属组。属主和属组之一都可以为空。如果没有属组，也就不需要冒号(:)了，但是带上冒号，会让 **chown** 命令把 *user* 的属组设为默认组。出于历史原因，*user.group* 这样的形式也能接受，但是由于用户名可以包括点，所以这种形式不够通用。

要改变一个文件的属组，您必须是该文件的属主而且属于目标属组的成员，或者必须是超级用户。您必须是超级用户才能改变文件的属主。

类似 **chmod**，**chown** 也提供了递归的**-R** 标志，能够改变一个目录以及在它之下所有文件的设定。例如，命令序列：

[12] 如果 **myprog** 是一个 shell 脚本，那么它需要同时打开读取和执行的权限。为了让解释程序运行这个脚本，这个脚本必须像文本文件一样被打开和读取。二进制文件是由内核直接执行的，因此不需要打开读取权限。

www.TopSage.com

```
# chmod 755 ~matt
# chown -R matt:staff ~matt
```

可以在复制了默认启动文件以后用来建立一个新用户的主目录。一定不要尝试采用类似下面的命令来 **chown** 新用户的点文件（.）：

```
# chown -R matt:staff ~matt/.*
```

这个模式将匹配~**matt/..**，导致其父目录的归属关系被修改，而且很可能还会造成其他用户主目录归属关系被修改。

传统的UNIX系统使用一个单独的命令（**chgrp**）去改变一个文件的属组。Linux也一样提供了 **chgrp** 命令。它的用法与 **chown** 命令基本相同，如果您觉得记住它更容易，不妨放心用吧。

5.5.7 umask：分配默认的权限

用户可以使用内建的 shell 命令 **umask** 来影响分配给新创建文件的默认权限。**umask** 用一个三位数字的八进制值形式来指定，这个值代表要“剥夺”的权限。当创建文件时，它的权限就设置为创建程序请求的任何权限去掉 **umask** 禁止的权限。因此 **umask** 的三位数字所对应的权限如表 5.6 所示。

例如，**umask 027** 允许属主具有所有权限，但禁止属组的写入权限，并且不允许其他任何用户有任何权限。默认的 **umask** 值是 022，它不允许属组和其他用户有写入权限。

表 5.6 umask 的作用代码

八进制	二进制	权限	八进制	二进制	权限
0	000	rwx	4	100	-wx
1	001	rw-	5	101	-w-
2	010	r-x	6	110	--x
3	011	r--	7	111	---

没有办法强制用户拥有某个特定的 **umask** 值，因为用户能够把这个值重设为他们想要的任何值。但是，在给新用户提供的**.cshrc** 和**.profile** 样本文件中，可以提供一个合适的默认值。有关启动文件的更多信息请参见第 6 章。

5.5.8 额外的标志

Linux 的 ext2fs 和 ext3fs 文件系统定义了一些补充属性，您可以打开它们来请求获得特殊的文件系统语义——“请求”是操作字，因为许多标志实际上还没有实现。例如，一个标志让文件只能追加（append），另一个标志让文件不可变动和不可删除。

既然这些标志不能用于 ext*系列以外的其他文件系统，所以 Linux 使用了特殊命令 **lsattr** 和 **chattr** 来查看和改变它们。表 5.7 列出了当前能用的标志（目前大约只占在手册页中提到的那些标志的 50%）。

表 5.7 ext2fs 和 ext3fs 的额外标志

标准	含义
A	从不更新访问时间（st_atime；出于性能原因）
a	只允许以追加模式写入（只有 root 能设置）
D	强制目录更新被同步写入

www.TopSage.com

大家网 TopSage.com

续表

标　准	含　义
d	不作备份——让 **dump** 忽略这个文件
i	让文件不可变动和不可删除（只有 root 能设置）
j	为数据变化和元数据都保留日志
S	强迫变动被同步写入（不作缓冲）

除了“不作备份”标志可能有所例外，还不清楚它们中间别的功能可以提供多少日用价值。“不可改变”和“仅限追加”标志多被认为能让系统对黑客或者恶意代码损坏的抵抗力更强。遗憾的是，它们会搞乱软件，并且只能防止还不知道使用 **chattr-ia** 的黑客的攻击[13]。实际经验已经表明，这些标志往往黑客用得比防黑客的还多。

启动同步写的 **S** 和 **D** 选项也应引起特别注意。由于它们迫使与某个文件和目录有关的所有文件系统页面都要随变动而立即写出到硬盘上，所以似乎能在发生崩溃的时候提供额外的保护，防止数据丢失。但是同步更新的操作次序不固定，而且已经知道这会把 **fsck** 搞糊涂，因此，可能导致受损文件系统的恢复更困难而不是更可靠。随 ext3fs 提供的文件系统日志机制通常是更好的选项。**j** 选项强制对特定文件的数据做日志，但这样会带来一些性能上的开销。

5.6 访问控制列表

9 位的属主/属组/其他人访问控制系统已得到证明是强大的，足以满足大多数管理方面的需求。虽然这种系统有明显的局限性，但是它却非常好地保持了 UNIX 简洁和可预测的传统（有人可能会说是“前传统”）。

事实上，在所有非 UNIX 的操作系统上都采用了一种实质上更为复杂的方式来管理对于文件的访问：访问控制列表（access control list），简称 ACL。ACL 不限长度，可以包含用于多个用户或者用户组的权限规定。更先进的系统能让系统管理员指定部分权限的集合或者否定方式的权限。有些系统还有继承特性，可以一次依靠多个 ACL 来指定访问权限。这些系统显然比传统的 UNIX 模型功能更强，但是对于系统管理员和软件开发人员来说，它们的复杂性也增加了一个数量级。

因为在 POSIX 规范中增加了 ACL，所以许多 UNIX 的变体也开始支持一种相当标准的 ACL 机制，这种机制和传统的 UNIX 9 位权限模式平行地发挥作用。在 Linux 下，ext2、ext3、ReiserFS、XFS 和 JFS 都支持 ACL。通常在默认情况下会禁用它们，在调用 **mount** 命令时加**-o acl** 选项就可以启用 ACL。参考 7.4.5 节了解有关 **mount** 命令的更多知识。

为了内容的完整性，我们在这里介绍 Linux 的 ACL 模型。但是不要被其华丽的外表所迷惑——ACL 不一定比传统的文件权限更好，在行的系统管理员应该在一定程度上谨慎地使用它们。不仅因为它们用起来复杂而且费事，而且它们在和 NFS、备份系统以及文本编辑器这样的程序联用的时候也会造成问题。ACL 在不断变化，所以随着时间推移在变得不可维护。

使用 ACL 表面上看起来最能让人接受的理由或许就是增强与其他操作系统的兼容性。具体来说，用来和 Windows 系统共享文件的 Samba 软件能够识别 ACL，实实在在地在 Linux 和 Windows 的 ACL 之间进行转换。参考 26.6 节了解有关 Samba 的更多信息。

[13] 在 20.6 节介绍的“权力（capability）”机制使得关闭这些附加标志位变得更困难，但是这项功能目前还没有广泛使用。

www.TopSage.com

5.6.1 ACL 概述

Linux 的 ACL 主要是对标准的 9 位权限模型直接进行扩展。读写和执行权限只是系统能够处理的权力（capability）。而像 setuid 和粘附位这样的功能还是通过传统的模式位专门处理。

ACL 可以按照用户和用户组的任意组合独立地设置 rwx 权限位。表 5.8 列出了 ACL 各组成项的样子。

表 5.8 ACL 中可以出现的项

格 式	举 例	权 限
user::*perms*	`user::rw-`	文件的属主
user:*username:perms*	`user:trent:rw-`	某个特定的用户
group: :*perms*	`group::r-x`	文件的属组
group:*groupname:perms*	`group:staff:rw-`	某个特定的组
other: :*perms*	`other::---`	所有其他人
mask: :*perms*	`mask::rwx`	除去属主和其他人之外的所有人[a]

a. mask 有点儿技巧，本节稍后介绍。

user 和 group 可以用名字或者 UID/GID 来标识。一个 ACL 能包含的组成项的数量随着文件系统实现的不同而不同，范围从 XFS 的最低 25 项，到 ReiserFS 和 JFS 的最多可以没有限制。ext2 和 ext3 文件系统可以有 32 项，对于在任何情况下都能管理来说，这或许是一个比较合理的上限。

getfacl 命令可以显示一个文件当前的 ACL，**setfacl** 命令可以修改或者设置文件当前的 ACL。**setfacl –b** *file* 命令可以清除 ACL，**setfacl –m** *aclspec file* 可以修改或者扩展 ACL，而 **setfacl –x** *aclspec file* 可以删除 ACL 中的特定项（使用-x 选项时省略权限说明部分）。只要在 *aclspec* 中用逗号分隔 ACL 项，就能够在其中包含多个 ACL 项。

带有 ACL 的文件仍然保留有它们原来的权限模式位，但是会自动在两套权限设置之间强制保持一致，使它们一定不发生冲突。下面的例子展示了 ACL 项随着 chmod 命令对权限模式位的修改而自动更新的情况：

```
$ touch /tmp/example
$ ls -l /tmp/example
-rw-rw-r--  1 garth  garth     0 Jun 14 15:57 /tmp/example
$ getfacl /tmp/example
getfacl: Removing leading '/' from absolute path names
# file: tmp/example
# owner: garth
# group: garth
user::rw-
group::rw-
other::r--
$ chmod 640 /tmp/example
$ getfacl --omit-header /tmp/example
user::rw-
group::r--
other::---
```

这种强制保持一致的做法让不知道 ACL 的老软件在有 ACL 的情况下也能很好地工作。不过有一点问题。虽然前面例子中 group::这个 ACL 项似乎对应了传统权限模式的中间一组权限位，但情况不一定总是这样。

www.TopSage.com

为了理解其中的原委，假定有个老程序清除了传统权限模式中所有三组权限位中的写权限位（例如，**chmod ugo-w** *file*）。这样做的目的很清楚，就是不让任何人能够写这个文件。但是如果产生的 ACL 将会是下面这个样子该怎么办？

```
user::r--
group::r--
group:staff:rw-
other::r--
```

从老程序的角度来看，文件不能修改，但是 staff 组中的任何成员实际上都对文件有写权。为了减少出现二义性和误会的机会，Linux 采用了下面的规则。

- ACL 项 user::和 other::定义为等价于传统文件模式中“owner（属主）”和“other（其他人）”的权限位。权限模式改变，对应的 ACL 项也改变。反之亦然。
- 在所有情况下，对于文件的属主以及没有以别的方式提到的用户，给它们赋予的有效访问权限就是在 user::和 other::这两个 ACL 项中分别指定的权限。
- 如果一个文件没有显示规定的 ACL，或者一个 ACL 只包含一个 user::、一个 group::和一个 other::项，那么这几个 ACL 项等同于传统的三组权限位。上面给出的 **getfacl** 的例子就属于这种情况。（这样的 ACL 称为“最小”ACL，实际上不需要按一个逻辑上独立的 ACL 来实现。）
- 在更复杂的 ACL 里，传统的组权限位对应于一个称为“mask”的特殊 ACL 项，而不是 group::那个 ACL 项。ACL 的 mask 项限制了 ACL 能够赋予所有有名字的用户、所有有名字的组以及默认组的访问权限。

换句话说，mask 规定了 ACL 能够给单个组和用户访问权限的上限。它在概念上类似于 **umask**，不同之处在于 ACL 的 mask 始终有效，而且规定的是允许的权限，而不是不允许的权限。给有名字的用户、有名字的组以及默认组的 ACL 项包括 mask 中没有出现的权限位，但是内核会忽略它们。

因此，传统的权限模式位不会描述不了 ACL 所允许的访问权限。而且从传统权限模式的组权限位中清除一位，也会清除 ACL mask 中对应的位，因而让除了该文件的属主以及处于“other”类的用户之外的所有人都没有该权限。

扩充前面例子里的 ACL，使它包括对某个特定用户和组的 ACL 项，**setfacl** 会自动提供一个合适的 mask：

```
$ setfacl -m user::r,user:trent:rw,group:admin:rw /tmp/example
$ ls -l /tmp/example
-r--rw----+ 1 garth  staff    0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
user::r--
user:trent:rw-
group::r--
group:admin:rw-
mask::rw-
other::---
```

如前所示，**setfacl** 命令产生了一个 mask，让 ACL 中赋予的所有权限都发生作用。如果不想手工设置 mask，可以在给 **setfacl** 命令的 ACL 项清单中包含这个 mask，或者使用 **-n** 选项避免 **setfacl** 命令重复产生它。

在尝试访问文件的时候，要把有效 UID 同该文件属主的 UID 进行比较。如果它们两者一样，那么 ACL 中的 user::项权限就决定了能否访问。否则，如果匹配某个特定于用户的 ACL 项，那么那个 ACL 项连同 ACL 的 mask 就一起决定了能否访问。如果没有特定于某个用户的 ACL 项，那么文件系统就尝试找到一个有效的组 ACL 项，能提供所请求的访问。这样的 ACL 也要和 ACL 的 mask 一起处

www.TopSage.com

理。如果没有找到匹配项，那么再用 other::这个 ACL 项。

如果在一个有 ACL 的文件上使用传统的 **chmod** 命令来控制组的访问权限，那么要注意修改只对 mask 有影响。继续用前面的例子来说明：

```
$ chmod 770 /tmp/example
$ ls -l /tmp/example
-rwxrwx---+ 1 garth  staff    0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
user::rwx
user:trent:rw-
group::r--
group:admin:rw-
mask::rwx
other::---
```

这里 **ls** 命令的输出有点误导性。尽管表面上看给组权限，但是没有人因为是组成员而有权执行该文件。为了让组有这样的权限，必须编辑 ACL 本身。

5.6.2 默认项

除了表 5.8 列出的 ACL 项之外，目录的 ACL 可以包括一个“default（默认）”项，加到在目录下新创建的文件和子目录的 ACL 上。子目录能接收主动和默认形式的 ACL 项。因此，最初的默认项最后会向下加给几层子目录。

在复制了默认项之后，父子 ACL 之间就不再有联系。如果父亲的默认项发生变化，不会反映到现有子目录的 ACL 上。

5.7 习题

E5.1 什么是 **umask**？创建一个 **umask**，对于属组或者其他人什么权限也不给。

E5.2 硬链接和符号（软）链接之间有什么不同？什么时候适用哪一种链接？

★E5.3 阅读**/etc/fstab** 文件的手册页。编写一行配置，在启动时自动安装一个 Windows NTFS 分区**/dev/hda1**。使用**/mnt/win_c** 作为安装点。

★E5.4 在安装一个 Linux 系统的时候，对硬盘驱动器分区用来给每个文件系统（**/var**、**/usr** 等）分配足够的空间很重要。“Foobar Linux”发行版本（译者注：这是作者为举例而随便杜撰的一个发行版本）使用下面的默认设置：

/	100MB
/var	50MB
/boot	10MB
<swap>	128MB
/usr	剩余的空间

在一个繁忙的服务器上采用这样的安排会有什么样的潜在问题？

★E5.5 为什么说将有些分区（比如**/var**、**/home** 和交换分区）放到一个和其他数据文件以及程序不同的单独驱动器上是个好主意？**/tmp** 呢？对于列出的每种文件系统给出有针对性的理由。

www.TopSage.com

★E5.6 写一个脚本，找到一个文件系统上所有的硬链接。

★E5.7 给出完成下面任务的命令：

a）把 **README** 的权限设置为对属主有读/写权，而其他每个人有读权；

b）在不需要改变（或者知道）目前权限的情况下，打开一个文件的 setuid 位；

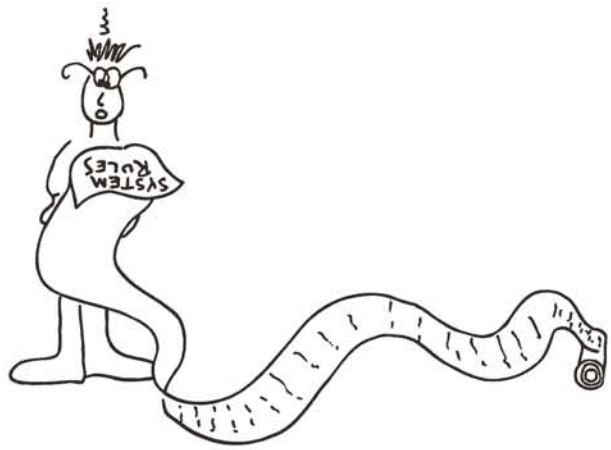
c）列出当前目录的内容，按照修改时间排序，把最近修改的文件放在最后；

d）将一个叫做 **shared** 的文件的属组从“user”改成“friends”。

www.TopSage.com

大家网
TopSage.com

第 6 章　添加新用户



添加和删除用户是大多数系统上例行的管理工作。这两项工作虽然简单，但是却枯燥无味。大多数系统管理员都做了工具自动完成增删用户的工作，然后让助手或者操作人员代为实际执行。

我们在今天看到，除了少到只有几个用户的分布式服务器，又重新出现了有成百上千登录账号的中央主机。为了管理好网络服务，为本地计算环境正确地配置账号，系统管理员需要对账号系统有透彻的了解。

账号安全也是系统安全的一项关键性的决定因素。不常用的账号，以及采用了很容易被猜中的口令的账号，都是黑客攻击的主要目标。即使使用系统的默认工具来添加和删除用户，懂得这些工具所做的各种幕后修改仍然是很重要的。

在这一章里，我们将首先研究自动工具所实现的下层模型，然后介绍工具本身（**useradd**、**userdel** 等）。系统提供的默认工具 **useradd** 实际用起来相当不错，应该能够满足站点的大多数需求。遗憾的是，**userdel** 却没有我们希望的那样好。

6.1　/etc/passwd 文件

/etc/passwd 文件是系统能够识别的用户的一份清单。在用户登录期间，系统查询这个文件，确定用户的 UID 并验证用户的口令。这个文件中的每一行都代表一个用户，它包含 7 个由冒号分隔的字段。

- 登录名。
- 经过加密的口令或者口令占位符（参见 6.1.2 节）。
- UID（用户 ID）号。
- 默认的 GID（组 ID）号。
- “GECOS”信息：全名、办公室、分机号、住宅电话。
- 主目录。
- 登录 shell。

www.TopSage.com

举几个例子，下面各行从语法上看，都是有效的**/etc/passwd** 文件条目：

```
root:lga5FjuGpZ2so:0:0:The  System,,x6096,:/:/bin/sh
jl:x:100:0:Jim  Lane,ECT8-3,,:/staff/jl:/bin/sh
dotty:$1$Ce8QpAQI$L.DvJEWiHlWetKTMLXFZO/:101:20::/home/dotty:/bin/tcsh
```

现如今，已经不能采用以纯文本显示加密口令的做法。随着硬件速度越来越快，几分钟时间就能“破解”这样的口令。所有版本的 UNIX 和 Linux 系统都能让用户把加密后的口令藏在一个不是所有人都可读的单独文件中。这就是隐蔽口令的机制，在大多数系统上，都默认采用它。

把隐蔽口令系统按一种对传统**/etc/passwd**（以其过去的形式）文件的扩展来解释的时候，它具有更多意义，所以我们等到 6.2 节再介绍它。本书 20.4 节更全面地讨论隐蔽口令所暗含的安全性。

/etc/passwd 中的内容经常通过一个像 NIS 或 LDAP 这样的数据库在多个系统间共享。如果想知道这方面的更多知识，请参看第 17 章的内容。

下面各小节中将更详细地讨论**/etc/passwd** 中的字段。

6.1.1 登录名

登录名（也叫做用户名）必须是唯一的，它的长度一般不超过 32 个字符。它们可以包括除冒号和换行（newline）之外的任何字符。如果使用了 NIS 系统，那么不管是哪种操作系统，登录名的长度都要限制在 8 个字符以内。参见 17.4 节，了解有关 NIS 系统的更多知识。

有些比较老的 UNIX 版本将允许登录名使用的字符限制为字母和数字，而且长度只能有 8 个字符。在异构的站点，要遵守最严格的限制。采取这样的策略会避免可能同老软件发生的冲突，确保用户能够在每台机器上都有相同的登录名。记住这一点，就因为今天拥有一个同构环境并不意味着明天也会是同构环境。

登录名要区分大小写，不过，RFC822 要求在邮件地址中忽略大小写。我们没有碰到因为登录名大小写混用而造成的任何问题，但是传统上采用全部为小写的登录名，这样做输入也比较容易。

登录名应该取得容易记住，因此随机排列的若干字母并不能组成很好的登录名。我们建议用户，即便是很不正式的单位，也应该避免使用“外号”和矫揉造作的昵称。它们确实没那么好玩，并且容易引起别人的嘲笑。在@hotmail.com 前加上 DarkLord 和 QTPie 就属于这类不严肃的名字。即使用户自己不自爱，也至少应该为整个站点的名誉想想。

既然登录名经常被用作电子邮件的地址，建立一套构成登录名的标准方法是非常有用的。这套方法应该让用户能够彼此猜出对方的登录名。用户的名、姓、首字母以及它们的一些组合都能够建立起一套合理的命名方案。

为选择登录名而确定的任何方案最终都会出现重名或者名字太长的问题，所以管理员有时不得不允许一些例外的情况出现。在出现很长名字的情况下，可以采用**/etc/mail/aliases** 文件，让两种格式的名字至少对于邮件来说是等价的。参考 18.4 节了解有关邮件别名的更多知识。

例如，假定用雇员的名字首字母和姓氏构成命名规范，由此，Brent Browning 的取名就是“bbrowning”，它有 9 个字符，所以太长了。取而代之的做法是，给这个用户分配“brentb”作为登录名，而把“bbrowning”留作 **aliases** 文件中的别名项。

```
bbrowning:  brentb
```

如果站点有一个全局的邮件别名文件，每个新的登录名都必须和这个文件中的任何别名不同。如果不是这样的话，邮件就会被发送给别名而不是新用户。

对于大型站点来说，常见的做法是实现一种采用用户全名的邮件寻址方案（例如，John.Q.Public @mysite.com），它把登录名对外部世界隐藏了起来。这是个不错的主意，但它确实还不足以不考虑前

www.TopSage.com

面提过的任何命名建议。如果管理员足够明智，那么登录名和用户实际的名字最好能有一种清楚的、可以推测出来的关系。

如果用户有一台以上的机器，那么从两种意义上来说，登录名都应该是唯一的。首先，用户应该在每一台机器上都有相同的登录名。这一规则主要是为了使系统管理员和用户都方便。

其次，某个特殊的登录名应该总是指同一个人。有些命令（例如，**ssh**）可以设置成根据远程用户的登录名来验证他们的身份。即使 `scott@boulder` 和 `scott@refuge` 是两个人，可如果系统没有设置好，其中一个人也可能不需要提供口令就能登录另一个人的账号。参考 20.9 节了解有关等价登录关系的更多知识。

经验还表明，重复的名字会造成邮件的混乱。邮件系统能十分清楚地知道哪个 `scott` 是哪个人，但是用户却经常把邮件发到了错误的地址。

6.1.2 加密的口令

在我们详细介绍口令之前，先提示一下：大多数系统实际上把加密后的口令放在**/etc/shadow** 而不是**/etc/passwd** 文件里。但不管实际情况下口令到底保存在哪儿，本节讨论的内容都适用。

/etc/passwd 文件以加密的形式保存口令。除非用户能够自己心算完成加密过程（我们想拜会您这位高人），否则必须通过使用 **passwd** 命令来设置这个字段的内容（如果使用的是 NIS 系统，那么就应该使用 **yppasswd** 命令），或者从另外一个账号中复制加密后的口令字符串。

如果通过手工编辑**/etc/passwd** 文件来创建一个新账号，那么要在加密口令字段添一个星号或者一个 x。这个星号可以防止在设置一个真实口令之前发生未经授权使用该账号的情况。千万不要让这个字段为空——那样会引入一个大大的安全漏洞，因为不要口令就能访问这个账号。即使使用了隐蔽口令的方式，注意一下**/etc/passwd** 文件里口令的安全仍然是明智的。因为不知道什么时候，有某个过时的程序或者脚本就会查这个文件，以此做出某种安全方面的决定[1]。

大多数 Linux 发行版本能识别多种口令加密算法，这些系统可以通过研究加密后的数据来判断出每个口令所用的加密算法。系统上的所有口令不必采用同一种加密方式。

大部分的 Linux 发行版本默认使用 MD5 算法来加密口令。MD5 比以前的 DES 标准算法加密效果稍好，MD5 算法可以使用任意长度的口令。口令越长也越安全——如果用户愿意在实际中采用它们的话。因为采用 MD5 没有损失，而且可能还有好处，所以我们建议在所有支持它的系统上都使用它。

不管加密前的口令有多长，加密后的口令长度都是固定的（MD5 算法的结果是 34 个字符，DES 则是 13 个字符）。在加密口令的过程中会随机掺入“盐（salt）”，这样一来，同一个口令可以对应许多不同的加密后的形式。如果两个用户碰巧选择了相同的口令，那么检查加密后的口令通常并不能发觉这种情况。MD5 口令很容易辨认，因为它们总是以1开头。



SUSE 默认给新口令用 Blowfish 加密算法。和 MD5 一样，这也是一种强加密算法，作为默认加密算法也很合理。不过，不能把 SUSE 的 Blowfish 口令复制到不是 SUSE 的系统上，因为只有 SUSE 能理解它们。Blowfish 的口令可以通过以$2a$开头来分辨。

6.1.3 UID 号

UID 是 32 位无符号整数。然而，因为和老系统之间的互操作性问题，我们建议在可能的情况下

[1] 我们的技术审稿人之一 Jon Corbet 建议说：“如果不知道什么时候要做出安全方面的决定，那么就已经面临麻烦了。系统管理员不应该对这样的事情感到突然。”

www.TopSage.com

将站点上的最大 UID 号限制为 32 767（最大的 16 位有符号整数）。

按照定义，root 的 UID 为 0。大多数系统还定义虚拟用户 bin、daemon 以及其他一些用户。习惯上将这些虚拟登录名放置在**/etc/passwd** 文件的开头部分，并且给予它们比较小的 UID 号。为了能够给将来可能添加的任何非真实用户提供足够的空间，我们建议从 500（或更大）开始分配真实用户的 UID 号。

让多个账号的 UID 都为 0 并不好。虽然有了多个具有不同 shell 和口令的 root 登录账户似乎比较方便，但这种设置会产生更多潜在的安全漏洞，并且需要保证多个登录账户的安全。如果用户需要有以 root 身份登录的替代方法，那么最好使用类似 **sudo** 这样的程序。参考 3.4.2 节了解有关 **sodu** 的更多信息。

即使某些 UID 的用户已经离开了您的公司，并且他们的账号已经被永久删除了，您也必须尽可能地避免再使用那些 UID。如果以后从备份中恢复文件，这种防范措施能够防止可能产生的混乱，因为在这个过程中是通过 UID，而不是通过登录名来确认用户身份的。

UID 在整个单位里应该保持唯一。也就是说，一个特定的 UID 在每台计算机上都应该指的是相同的登录名和同一个用户。如果不能保持 UID 的独特性，那么在诸如 NFS 这样的系统中就可能造成一些安全方面的问题，而且当用户从一个工作组移到另外一个工作组中时还会造成混乱[2]。如果想知道关于 NFS 的更多信息，请参考第 16 章。

当很多组计算机是由不同的人或者单位来管理时，保持各个 UID 的唯一性就会变得困难起来。这些问题既是技术上的也是行政上的。最好的解决办法就是建立一个包含每个用户的记录并落实其唯一性的中央数据库（我们采用了自己为解决这个问题而建立的数据库）。更为简单的方案就是为单位内的每一组用户都分配一定范围的 UID，并且让每组管理自己的 UID 范围。这种方法保证了 UID 空间的互相独立（当使用 NFS 来共享文件系统时，这是一个必须满足的要求），但没有解决平行存在“唯一”登录名这个问题。LDAP 也成为一种流行的 UID 管理工具。

6.1.4 默认的 GID 号

和 UID 类似，组的 ID 是一个 32 位的整数。GID 0 是给名为“root”的组保留的。GID 1 通常指的是名为“bin”的组，GID2 指的是“daemon”组。

组在**/etc/group** 文件中定义，**/etc/passwd** 中的 GID 字段在登录时提供默认（或者“有效”）GID。在判断访问权限的时候并不特别看重默认 GID[3]，而它只是在创建新文件和新目录的时候才起作用。新文件一般由用户的有效组所有。不过，在已经设置了 setgid 位（02000）的目录里，以及在用 **grpid** 选项安装的文件系统上，新文件默认为父目录的组所有。参考 5.5.2 节了解有关 setgid 目录的更多知识。

6.1.5 GECOS 字段[4]

GECOS 字段通常用来记录每个用户的个人信息。它的语法定义不明确。GECOS 字段最初是用来保存登录信息的，在把批处理作业从贝尔实验室中的 UNIX 系统传送到运行 GECOS（代表 General Electric Comprehensive Operating System）的大型机时需要这些登录信息，现如今只留下了 GECOS 这个名字。有几个程序把 GECOS 字段中的“&”扩展为用户的登录名，这样省去了输入。**finger** 和 **sendmail**

[2] 另一个和 NFS 有关的问题是“nobody”这个 UID，传统上用这个 UID 限制远程 root 用户访问。参见 16.1 节了解细节。

[3] Linux 在计算访问权限时，所有的组都要考虑到。2.6.4 之前的内核允许用户同时属于至多 32 个组，之后版本的内核没有限制。

[4] 在 Honeywell 接管了 GE 的计算机业务之后，GECOS 就变为 GCOS，如今这两种拼法都存在。

www.TopSage.com

这两个程序都会执行上述扩展，但是许多程序不会执行。所以最好不要依靠这一特性。

尽管可以使用自己喜欢的任何格式约定，但 **finger** 按如下的顺序来解释那些用逗号隔开的 GECOS 项：

- 全名（经常是唯一使用的字段）；
- 办公室房间号和楼名；
- 办公室电话分机号；
- 家中的电话号码。

chfn 命令可以让用户改变他们自己的 GECOS 信息。**chfn** 对于保持诸如电话号码之类信息的更新非常有用，但它也可能被误用：一个用户可以把那些信息修改为一些淫秽的或者错误的信息。大多数大学校园就禁止使用 **chfn** 命令。GECOS 信息非常适合用 LDAP 管理。

6.1.6 主目录

当用户登录之后，他们的 shell 就进入到自己的主目录中。如果在登录时找不到用户的主目录，那么系统会显示诸如“no home directory（无主目录）”这样的消息[5]。如果把**/etc/login.defs** 中的 DEFAULT_HOME 设为 no，那么就会禁止没有主目录的用户登录；否则会允许继续登录，并且让用户处在根目录下。

注意，如果主目录是通过 NFS 安装的，那么它们在服务器或者网络出现问题时可能不能使用。

6.1.7 登录 shell

虽然登录 shell 一般是像 Bourne shell 或者 C shell（**/bin/sh** 或者**/bin/csh**）那样的命令解释程序，但它可以是任何程序。**bash** 是默认的 shell，如果**/etc/passwd** 没有指定一个登录 shell，那么就用它。在 Linux 系统上，**sh** 和 **csh** 实际上分别链接到了 **bash**（GNU 的“Bourne again” shell）和 **tcsh**（C shell 的一个超集）上。大多数发行版本还提供一个公共域版本的 Korn shell，**ksh**。

用户可以使用 **chsh** 命令来改变自己所用的 shell。文件**/etc/shells** 中包含了 **chsh** 命令允许用户选择使用的“有效” shell 的列表。SUSE 强制使用这个列表，而 Red Hat 只是在所选择的 shell 不在列表中时发出警告。如果向 **shells** 文件中增加项，那么一定要使用绝对路径，因为 **chsh** 和其他一些程序都要求使用绝对路径。

6.2 /etc/shadow 文件

只有超级用户才能读**/etc/shadow** 文件，它用于保护加密口令的安全，使口令不被爱窥探的眼睛看到。它还提供了无法从**/etc/passwd** 得到的账号信息。有些发行版本的标准是采用隐藏口令系统，而在其他发行版本上则配置为一种可选软件包。即使隐藏口令是可选的软件包，把它们当作是标准系统来用也是个不错的主意。

在使用了隐蔽口令系统的情况下，**/etc/passwd** 文件中原来的口令字段一定要包含一个 **x**。

shadow 文件并不是 **passwd** 文件的超集，而 **passwd** 文件也不是从它产生的，管理员必须两个文件都维护（或者使用诸如 **useradd** 这样的工具来自行维护）。和**/etc/passwd** 类似，**/etc/shadow** 文件也

[5] 当用户在控制台或者终端上登录的时候会出现这则消息，但是当用户通过图形管理器（比如 **xdm**、**gdm** 或者 **kdm**）登录的时候则不会显示。用户不但看不到消息，而且一般会立即退出登录，因为显示管理器不能向正确的目录（例如，**~/.gnome**）写信息。

www.TopSage.com

大家网 TopSage.com

是给每个用户一行，每行有 9 个用冒号分隔的字段：

- 登录名；
- 加密后的口令；
- 上次修改口令的时间；
- 两次修改口令之间最少的天数；
- 两次修改口令之间最多的天数；
- 提前多少天警告用户口令即将过期；
- 在口令过期之后多少天禁用账号；
- 账号过期的日期；
- 保留字段，目前为空。

唯独要求非空的字段是用户名和口令。**/etc/shadow** 中指定的绝对日期是从 1970 年 1 月 1 号至今的天数（不是秒数），这不是 UNIX 系统上估算时间的标准方法。幸运的是，可以使用 **usermod** 程序来设置过期字段。参考 6.7 节了解有关 **usermod** 的更多知识。

:一个典型的 **shadow**（隐藏口令文件）看上去类似于：

```
millert:$1$buJ6v3Ch$BwLIoF5eaCh9Nv.OEzD3T0:13348:0:180:14::14974:
```

下面是对每个字段更完整的描述。

- 登录名和**/etc/passwd** 中的登录名相同。这个字段只不过是将用户的 **passwd** 和 **shadow** 中的记录联系在了一起。
- 加密的口令在概念上和使用上都和先前存储在**/etc/passwd** 中的口令完全相同。
- 上次修改字段指出了用户口令上一次被修改的时间。这个字段通常由 **passwd** 填写。
- 第 4 个字段设置了两次口令修改之间必须间隔的最少天数。其用意是迫使口令的修改可信，即防止用户在按要求修改了口令之后马上又改回一个自己熟悉的口令。不过我们认为一旦发生安全入侵，此项设置就可能有些危险。所以我们建议将这个字段设置为 0。
- 第 5 个字段设置了两次口令修改之间所允许间隔的最多天数。这项功能让管理员加强了口令的时效性。如果想知道更多信息，请参看第 20.5.2 节中的内容。实际的最多天数是最多天数字段中的天数和第 7 个字段（不活动字段）中的天数之和。
- 第 6 个字段设置了在口令作废之前多少天，**login** 程序应该开始提醒用户口令即将作废。
- 第 7 个字段指定了在达到最大的口令作废天数之后，登录账号作废之前必须等待的天数。这项功能的确切用意并不清楚。
- 第 8 个字段指定了用户账号作废的天数（从 1970 年 1 月 1 日开始的天数）。如果管理员没有重新设置这个字段的值，那么过了这个日期之后该用户也许就不能用此账号登录了。如果这个字段的值为空，那么该账号永远都不会作废。
- 第 9 个字段被保留供将来使用。

既然我们已经知道了每个字段的意义，那么就让我们再看看下面这个例子：

```
millert:$1$buJ6v3Ch$BwLIoF5eaCh9Nv.OEzD3T0:13348:0:180:14::14974:
```

在这个例子中，用户 millert 上次修改他自己的口令的时间是 2006 年 6 月 18 日。口令必须在 180 天内再次进行修改，在这个阶段的最后两个星期里，millert 将会收到必须修改口令的警告。该账号将在 2010 年 12 月 31 日过期。

您可以使用工具 **pwconv** 让 **shadow** 文件的内容和 **passwd** 文件的内容保持一致，补上任何新加的用户，删除不在 **passwd** 中列出的用户。**pwconv** 会用**/etc/login.defs** 里指定的默认值填充 **shadow** 文件里大多数的隐蔽参数。

www.TopSage.com

下面这个例子显示了 **login.defs** 文件的格式。注释很好地解释了各种参数的含义。这个例子取自一个 Fedora 系统；就像可以指定的参数差别很大一样，各个发行版本上该文件的内容大相径庭。

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail

# QMAIL_DIR       Maildir
MAIL_DIR          /var/spool/mail
# MAIL_FILE       .mail

# Password aging controls:
# PASS_MAX_DAYS  Maximum # of days a password may be used.
# PASS_MIN_DAYS  Minimum # of days allowed between password changes.
# PASS_MIN_LEN   Minimum acceptable password length.
# PASS_WARN_AGE  Number of days warning given before a password expires.

PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

# Min/max values for automatic uid selection in useradd

UID_MIN                 500
UID_MAX                 60000

# Min/max values for automatic gid selection in groupadd

GID_MIN                 500
GID_MAX                 60000

# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).

# USERDEL_CMD     /usr/sbin/userdel_local

# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.

CREATE_HOME     yes
```

6.3 /etc/group 文件

/etc/group 文件包含了 UNIX 组的名称和每个组中成员的列表。例如：

```
wheel:x:10:trent,net,evi,garth,lynda,boggs,millert
csstaff:*:100:lloyd,evi
student:*:200:dotty
```

每一行都代表了一个组，其中包含有 4 个字段：

- 组名；
- 加密的口令或者包含一个 **x**，表明有一个 **gshadow** 文件；

www.TopSage.com

- GID 号；
- 成员列表，彼此用逗号隔开（注意不要加空格）。

和**/etc/passwd** 里的格式一样，字段之间用冒号隔开。虽然 Linux 实际上并不限制组名必须是 8 个字符长或者更短，但是为了兼容起见，还是应该遵守这一限制。虽然可以输入一个组的口令（通过使用 **newgrp** 命令让不属于某个组的用户变为属于该组），但我们很少这么做[6]。大多数站点都在口令字段中放置星号（*），但如果您愿意的话，将口令字段留为空白也是很安全的。除非用户已经列为该组的成员，否则 **newgrp** 命令没有口令是不能把这个用户变为该组成员的。我们举例的发行版本都配置有**/etc/gshadow** 文件，它的概念和**/etc/shadow** 文件类似，但是却不重要得多（组的口令很少使用）。

与用户名和 UID 一样，组名和 GID 也应该在通过网络文件系统共享文件的计算机之间保持一致。在异构环境中保持一致性会比较难，因为不同的操作系统对于相同的组名使用的是不同的 GID。我们已经发现，处理这个问题的最好方法就是避免将一个系统组用作用户的默认登录组。

如果一个用户默认属于**/etc/passwd** 中某个特定的组，但是根据**/etc/group** 文件的内容，却又不在那个组里，那么就要按**/etc/passwd** 的内容对待。在登录时，用户该是哪些组的成员，实际上是 **passwd** 和 **group** 文件中找到的组的并集。不过，最好保持两个文件一致。

为了避免与厂商提供的 GID 发生冲突，我们建议从 GID 500 开始分配本地组。

UNIX 的传统总是把新用户加入到一个代表其普通类型的组里，比如 students 或者 finance 组。但值得说明的是，这样的约定增加了用户因为不经意地设置权限而能够彼此读取文件的可能性，即使这并非文件属主的本意。为了避免这个问题，我们建议您为每个用户创建一个唯一的组。您可以让用户和组的名字一样，还可以让 GID 和 UID 一样。

用户的个人组应该只包含用户本人。如果您想要用分组机制让用户共享文件，那么应该为此目的单独建组。个人组背后的思想本身不是为了妨碍组本身的使用——而只是为了给每个用户建立一个更严格的默认组，从而让文件不会被意外共享。



除了 SUSE 之外，我们举例的其他发行版本上的工具程序 **useradd** 默认把用户放入它自己的组。

6.4 添加用户

在一个企业、政府或者教育站点上为新用户创建账号之前，该用户在本地用户协议和政策声明的副本上签字并注明日期是非常重要的（什么？！您还没有用户协议和政策声明？想知道为什么需要它们，以及其中应该包含哪些内容，请参看 30.11 节，了解更多信息）。

用户没有特别的理由会想着要签署一份政策协议，所以如果您还有一些影响力的话，那么就应该保证这些签名的安全。我们发现，在一个账号创建之后还保证这些签名协议的安全要花费更大的气力。如果您的规程考虑到了这一点，那么最好是在创建账号之前就做好文书工作。

通常，添加一个新用户的过程包含系统要求的 4 个步骤，两个步骤用来为新用户建立一个有用的环境，还有其他几个步骤用来为管理员自己提供更多便利。

系统所需的步骤：

- 编辑 **passwd** 和 **shadow** 文件，定义用户账号；
- 把用户添加到**/etc/group** 文件里；

[6] 我们知道有人想要在 Linux 下用 **newgrp** 命令的唯一原因就是为新建的文件设置默认的属组。

www.TopSage.com

- 设置一个初始口令；
- 创建用户主目录，用 **chown** 和 **chmod** 命令改变用户主目录的属主和属性。

为用户所进行的步骤：

- 将默认的启动文件复制到用户的主目录中；
- 设置用户的邮件主目录并建立邮件别名。

为管理员准备的步骤：

- 核实账号是否设置正确；
- 将用户的联系信息和账号状态加入数据库。

从 6.7 节开始，我们将讨论 **useradd** 及其同类命令，它们能够替您执行上述步骤中的一些，但是在下面几小节里，我们将会介绍手工操作时要执行的步骤。这些是主要步骤，所以您可以看到系统提供的工具在做什么。在现实里，一般比较可取的做法是运行 **useradd** 或者您自己写的某种类似脚本（更快且更不容易出错）。

您必须以 root 的身份来执行每一步骤，或者也可以使用诸如 **sudo** 这样让您以 root 身份来执行命令的程序。参见 3.4.2 节了解有关 **sudo** 的更多信息。

6.4.1 编辑 passwd 和 shadow 文件

为了安全地编辑 **passwd** 文件，可以运行 **vipw** 命令启动文本编辑器去打开它的一个副本。默认的文本编辑器是 **vi**，但也可以通过设置环境变量 EDITOR 的值来指定不同的文本编辑器。临时编辑文件的存在起着锁定的作用，**vipw** 每次只允许一个人编辑 **passwd** 文件，它防止用户在 **passwd** 文件被编辑的时候修改他们的口令。当文本编辑器终止时，**vipw** 将用编辑过的副本替换原来的 **passwd** 文件。

在 Fedora 和 RHEL 系统上，**vipw** 命令会自动询问编辑完 **passwd** 文件之后是否再编辑 **shadow** 文件。SUSE、Debian 和 Ubuntu 的系统要使用 **vipw –s** 来实现上述功能。

例如，将下面的一行添加到**/etc/passwd** 文件中，它将定义一个名为“tyler”的账号：

```
tyler:x:2422:2422:Tyler Stevens, ECEE 3-27, x7919,:/home/tyler:/bin/sh
```

我们还可以在**/etc/shadow** 文件里加入对应的一项：

```
tyler:*: : : : : :14974:
```

shadow 文件中用于“tyler”用户的一行没有加密口令或者口令的时效性信息，账号的作废时间被设为 2010 年 12 月 31 日。

6.4.2 编辑/etc/group 文件

我们应该在**/etc/group** 文件里给 tyler 建立个人的用户组，我们仍然把这个用户组叫做“tyler”。它的组 ID 应该是 GID 2422，和 tyler 的 UID 2422 相匹配。这是我们在 **passwd** 文件里给他分配的默认 GID[7]。

```
tyler::2422:tyler
```

严格地讲，无论 tyler 是否在**/etc/group** 文件中列出，他都将在组 2422 中，因为在 **passwd** 中的记

[7] 起的名字和分配的编号都纯粹按习惯，参见 6.3.1 节。

www.TopSage.com

录早已经给出了他的成员身份。内核实际上并不关心**/etc/passwd** 或**/etc/group** 文件中有哪些内容；它只关心那些原始的 UID 号和 GID 号。在 **group** 文件里保存个人用户组的主要目的，是为了让 **ls** 这样的命令能够正确地显示出这些组的名字。当然，有一个权威的列表列出创建的用户组及其成员总是不错的。

如果我们想把 tyler 分配给其他组，只要在**/etc/group** 文件中把他的登录名加入到其他组里就行了。

6.4.3 设置初始口令

root 能够使用 **passwd** 命令修改任何用户的口令：

```
# passwd user
```

或者

```
$ sudo passwd user
```

passwd 提示用户输入一个新口令，并且要求重复输入一次该口令。如果选择一个全是小写字符的不长的口令，或是其他看上去明显不合适的口令，那么 **passwd** 就会发出抱怨，并要求使用复杂一些的口令。大多数 Linux 系统还把预期的口令同一个字典做对比，以增强安全性。20.5.1 节给出了选择好口令的规则。

Don Libes 的 **expect** 软件包所带的工具 **mkpasswd** 能够很容易地为新用户产生随机口令。虽然随机口令[8]很难记住，但无论好坏，分一个随机口令的做法能“迫使”新用户立即改变他们的口令。不要把 **expect** 的 **mkpasswd** 命令和标准的 **mkpasswd** 命令混淆起来，后者只是将一个给定的字符串加密为口令。

千万不要让一个新账号——或者任何能够访问 shell 的账号——没有口令。

6.4.4 创建用户的主目录

系统管理员创建的任何目录最初都由 root 所有，所以必须使用 **chown** 命令修改它的属主和属组。下面这一系列命令将为我们举例的用户创建一个合适的主目录。

```
# mkdir /home/tyler
# chown tyler:staff /home/tyler
# chmod 700 /home/tyler
```

6.4.5 复制默认启动文件

通过将配置文件放置到用户的主目录中，可以对一些命令和工具进行定制。启动文件一般都是以圆点（.）开头，以字母 **rc**（“run command”的缩写）结尾，这是 CTSS 操作系统留下的“遗迹”。除非 **ls** 使用**-a** 选项，否则开头的圆点让 **ls** 在显示的目录列表中不显示这些文件，这些文件被认为是“不感兴趣的”。表 6.1 中列出了一些常见的启动文件。

[8] 生成的口令不是真正随机的，但还是伪随机的。如果伪随机生成的一系列口令中有一个或者几个口令被破解了，那么就有可能通过逆向工程处理这一系列口令，找出更多的口令。这种情况虽有可能，但是现实中不大会出现。我们对这样的风险大可以放心。

www.TopSage.com

表 6.1 常见的启动文件及其用途

命 令	文 件 名	典型的用途
csh/tcsh	.login	设置终端类型（如果需要） 设置 biff 和 mesg 的选项
	.cshrc	设置环境变量 设置命令别名 设置搜索路径 设置控制权限的 umask 值 设置搜索文件名的 cdpath 设置提示符（prompt）、历史（history）和 savehist 变量
bash[a]	.bashrc .bash_profile	和.cshrc 类似，用于 bash 和.login 类似，用于 bash
vim	.vimrc	设置 vim 编辑器的选项
emacs	.emacs	设置 emacs 编辑器的选项 设置 emacs 键绑定
mail/mailx	.mailrc	定义个人邮件别名 设置邮件阅读器选项
xrdb[b]	.Xdefaults	指定 X11 配置：字体、颜色等
startx[b]	.xinitrc .Xclients	指定初始的 X11 环境 指定初始的 X11 环境（RHEL、Fedora）
xdm[b]	.xsession	指定初始的 X11 环境

a. bash 也会模拟 sh，读取.profile 或者/etc/profile。
b. X 窗口的配置细节随实现和使用的窗口管理器不同而不同，参见第 22 章了解详情。

如果还没有一组不错的默认启动文件，那么/usr/local/lib/skel 是一个放置这些文件的合理位置。复制一些文件用来作为起点，并且用文本编辑器来修改它们。您也许希望用厂商在/etc/skel 下提供的文件，把它们作为修改的起点。还要保证给新用户的 umask 设置一个合理的值（我们建议采用 077、027 或者 022，具体是哪一个取决于站点的友好性和规模）。

根据用户 shell 的不同，/etc 可能包含在整个系统范围内起作用的启动文件，这些文件在用户自己的启动文件之前执行。例如，bash 会在处理~/.bash_profile 之前先读取/etc/profile 文件。对于其他 shell，请参看它们的手册页以了解详细情况。

对于 shell 的系统级启动文件来说，查看/etc/profile.d 目录获得要执行的附加配置片段，这已经成为比较常见的做法。这一习惯为软件包指定 shell 级的默认配置提供了一条清楚的途径。例如，Fedora 和 RHEL 系统上的/etc/profile.d/colorls.*文件负责 ls 命令在系统上输出的颜色。（删除这些文件不会有危险。）

为新用户 tyler 安装启动文件则需采用类似下面的命令：

```
# cp /usr/local/lib/skel/[a-zA-Z]* ~tyler
# chown tyler:staff ~tyler/[a-zA-Z]*
# chmod 600 ~tyler/[a-zA-Z]*
```

请注意，不能使用：

```
# chown tyler:staff ~tyler/.*
```

因为那样的话，tyler 不但是自己文件的属主，同时还是其父目录“..”（/home）的属主。这是一个非常常见而且非常危险的系统管理错误。

6.4.6 设置用户的邮件主目录

对于每个用户而言，只在一台计算机上接收邮件是非常方便的。这种方案经常可以通过中央邮件

www.TopSage.com

服务器上全局别名文件/etc/mail/aliases 中的一项或者 sendmail 中的 userDB 项来实现。请参看第 18 章中关于电子邮件的全面介绍，实现邮件主目录的各种方法在第 18.3.2 节中讨论。

6.4.7 验证新的登录

为了验证一个新账号是否已经正确配置好了，应该首先退出登录，然后再以新用户身份登录，并且执行以下这些命令：

```
$ pwd          /*  核实主目录  */
$ ls-la        /*  检查启动文件的属主和属组  */
```

系统管理员必须通知新用户他们的登录名和初始口令。许多站点用电子邮件来发送这些消息，但是出于安全原因，这样做通常并不是个好主意。因为这样做的话，新用户的账号可能在登录之前就被攻破，并且安装了后门。如果您有一些针对本地情况的文档说明，那么此刻也是一个很好的时机，可以向用户推荐它们。

如果站点要求用户签署一份书面的政策协定或者相关的使用规定，那么应该确保在提供该账号之前完成这一步工作。这项检查可以防止疏漏，并且可以加强以后可能会加入的条约的法律基础。参考 30.11.2 节了解有关用户书面协议的更多信息。还要记着提醒新用户立即修改他们的口令。如果您愿意，您可以通过设定口令在一段很短的时间内作废来强制实施这一要求。另一种可行的做法是利用一个脚本检查新用户的信息，确认 shadow 文件中保存的加密口令已经变了[9]。

6.4.8 记录用户的状态和联系信息

系统管理员如果认识所有的用户，那么很容易就能追查到是谁在使用系统，为什么使用系统。但是如果您管理的用户群庞大而且多变，那么就需要更为正式的渠道来跟踪账号的使用情况。维护联系信息和账号状态的一个数据库，能够帮助管理员在忘记加过某些用户的时候，判断出谁是谁，他们为什么会有一个账号。最好手头能保存完整的联系信息，在出现问题或者用户行为不端的情况下可以联系到用户。

6.5 删除用户

当某个用户离开您的单位时，应该从系统中删除该用户的登录账号和文件。这个过程包括删除由您或者您的 useradd 程序所添加的对登录名的所有引用。如果想要手工删除一个用户，可能需要对照下面的步骤来做：

- 将用户从所有本地用户数据库或者电话簿中删除；
- 将用户从 aliases 文件中删除，或者添加一个转发地址；
- 删除用户的 crontab 文件和所有挂起的 at 作业；
- 终止所有仍在运行的用户进程；
- 将用户从 passwd、shadow、group 和 gshadow 文件中删除；
- 删除用户的主目录；

[9] 因为相同的口令可以有许多种加密后的样子，所以这种方法只能验证用户重新设置了口令，但是不能真正确定已经改为不同的口令。除非保留用户以前设置的所有口令值的数据库，否则实际中间没有办法迫使用户真正改变他们的口令。

www.TopSage.com

- 删除用户的邮件存储文件。

在删除用户的主目录之前，应该确保先重新安置其他用户需要的任何文件。因为您往往不能确认那些文件是什么文件，所以在删除用户主目录和邮件存储文件前，给它们做个备份总是一个好的主意。

一旦删除了某个用户，就应该验证该用户的旧 UID 在系统中不再拥有文件。为了找到这些残存的文件，您可以使用带**-nouser** 参数的 **find** 命令。因为您如果不小心，**find** 就会找到网络服务器上去，所以一般最好用**-xdev** 逐个检查文件系统：

```
# find filesystem -xdev -nouser
```

如果您的单位给用户一人分配一台工作站，那么一般最简单也是效率最高的做法是，照一个样板重新安装整个系统，再把系统交给新用户。不过，在重装系统之前，最好备份系统硬盘上的所有本地文件，以防将来还会用得上它们。

6.6 禁止登录

有时候必须临时禁止一个用户登录。直接的做法是在加密的口令前面添加一个星号或者其他字符，这样就可以让大多数通过口令的访问不能登录，因为解密口令不能得到任何有意义的东西。不过，像 **ssh** 这样不必检查系统口令的命令仍然会起作用。



在我们所举的发行版本中，除了 SUSE 之外，都可以用 **usermod –L** *user* 和 **usermod –U** *user* 命令对口令上锁和解锁。

另外一种（或许更为安全）能够达到类似结果的做法是用一个程序替换用户的 shell。这个程序会显示一条信息来解释为什么禁止登录，并且提供纠正这种情况的指导。这个伪 shell 不应该列在 **/etc/shells** 中，很多提供对系统非登录访问的守护进程（如 **ftpd**）会检查用户的登录 shell 是否列在 **/etc/shells** 中，如果没有列出，守护进程就会拒绝访问（这正是您所需要的动作）。遗憾的是，如果您的站点一般通过窗口系统登录的话，用户可能看不到这则消息。

不过，使用这种方法来禁止登录会带来另一个问题。默认情况下，**sendmail** 不会把邮件投递给那些 shell 没有列在**/etc/shells** 中的用户。即使收件人不能立刻读到邮件，干涉邮件投递的流程往往也不是个好主意。您可以通过把一个名为**/SENDMAIL/ANY/SHELL/**的虚拟 shell 添加到**/etc/shells** 文件中，使得 **sendmail** 的默认动作失效（尽管这样做有可能会有不必要的副作用）。

6.7 账号管理工具

useradd 命令把用户添加到 **passwd** 文件中去（如果可以的话，还会添加到 **shadow** 文件中去）。它提供了一个命令行驱动的接口，很容易通过手工运行或从自制的 **adduser** 脚本调用。**usermod** 命令可以改变已有用户在 **passwd** 文件中的条目。**userdel** 命令可以将用户从系统中删除，还可以删除用户的主目录。**groupadd**、**groupmod** 和 **groupdel** 命令也可以对**/etc/group** 文件进行操作。

例如，为了用 **useradd** 命令（使用系统的默认设置）创建新用户 hilbert，只需运行：

```
# useradd hilbert
```

这条命令将在**/etc/passwd** 文件中创建下面的项。需说明的是，**useradd** 在口令字段中放置了一个星号，直到您指派一个真实的口令之前，它都能有效地禁用该账号。

www.TopSage.com

```
hilbert:*:105:20::/home/hilbert:/bin/bash
```



出于某种原因，SUSE 采用了一套类似但却独立开发的用户和用户组管理命令。这些命令的名字一样，但是在某些选项的含义和默认操作上有细微的差别。例如，大多数发行版本如果不在命令里多做指定，那么就会为新用户创建一个新组。SUSE 的 **useradd** 命令则把新用户加到组 100（在默认配置下，SUSE 还把新用户都加到“video”和“dialout”这两个组里）。

当给出其他参数的时候，**useradd** 往往会更有用。在下面的例子中，我们指定 hilbert 的主属组为“faculty”，而且还应该把他加入到“famous”组里。我们也重新指定了默认的主目录位置，如果该目录不存在，还要让 **useradd** 创建它。

```
# useradd -c "David Hilbert" -d /home/math/hilbert -g faculty -G famous -m
    -s /bin/sh hilbert
```

这条命令在 **passwd** 文件中创建了这么一条：

```
hilbert:x:1005:30:David  Hilbert:/home/math/hilbert:/bin/sh
```

（指派的 UID 比系统上最大的 UID 大 1）在 **shadow** 文件中相应的项为：

```
hilbert:!:11508:0:99999:7:0::
```

它还在**/etc/group** 中将 hilbert 添加到“faculty”组和“famous”组中，还创建**/home/math/hilbert** 目录，并按照**/etc/skel** 的内容在这个目录下增加若干初始配置文件。

在我们所举的所有发行版本中（除了 SUSE），都可以使用 **useradd -D** 命令决定 **useradd** 的默认配置，也可以将**-D** 标志和其他参数连用来设定那些默认配置。即使是在 SUSE 上，默认配置也仍然保存在文件**/etc/default/useradd** 里，如果愿意，可以直接编辑它。

usermod 命令用来修改已经存在的账号，并且使用了很多和 **useradd** 命令一样的标志。例如，我们能够使用下面的命令将 hilbert 账号的作废日期设为 2007 年 7 月 4 日：

```
# usermod -e 2007-07-04 hilbert
```

userdel 命令删除用户账号，而且取消 **useradd** 命令所做的全部修改。要删除 hilbert 这个账号，我们可以使用下面的命令：

```
# userdel hilbert
```

这条命令删除了 **passwd**、**shadow** 和 **group** 文件中与 hilbert 相关的资料。默认情况下，它不会删除 hilbert 的主目录[10]。**-r** 选项让 **userdel** 也删除用户的主目录，但是即便尽 **userdel** 最大能力，它还是只完成了“删除用户任务”清单中的最后三项。

虽然 **useradd** 和 **userdel** 命令非常方便，但是它们往往不足以实现一个站点所有的本地政策。别犹豫了，还是编写您自己的 **adduser** 和 **rmuser** 脚本吧，大多数比较大的站点都会这么做（Perl 通常是完成这项任务的合适工具）。您自己写的脚本可以调用标准的工具来实现部分任务。

6.8 习题

E6.1 怎样决定用户的默认组？怎样改变它？

[10] 在我们的站点上，我们一般会把已删用户的主目录保留几周。如果被删用户又要恢复，或者别的用户需要访问被删用户用过的文件，那么这一政策会把需要从磁带备份恢复数据的情况减至最低。

www.TopSage.com

E6.2 说明下面这几个 umask 值的区别：077、027、022 和 755。为了让这些值的其中之一成为用于新用户的全局默认值，该怎样做？您可以给用户施加一个标准的 **umask** 吗？

E6.3 隐蔽口令文件的目的是什么？

★E6.4 列出不用 **useradd** 程序，给系统增加一个用户所需的步骤。对于您本地的环境来说，额外还需要什么步骤？

★E6.5 确定您的站点上用于新用户的命名约定。采用什么规则？如何保持唯一性？您会想到有什么缺点吗？如何删除用户？

★★E6.6 找到一份学生名单（可能要从本地的在线电话簿找），用它作为脚本的输入，按照您站点上的命名约定构成登录名。在您遇到冲突之前，可以受理多少用户？总共有多少个冲突？使用这些数据来评定您站点的命名约定，并给出改进建议。

★★E6.7 编写一个脚本，帮助监视**/etc/passwd** 文件是否正常（除非您动动脑筋，否则 b 和 e 需要有超级用户权限）。

a）找出有 UID 0 的所有项；

b）找出没有口令的所有项（需要**/etc/shadow** 文件）；

c）找出有重复 UID 的所有项；

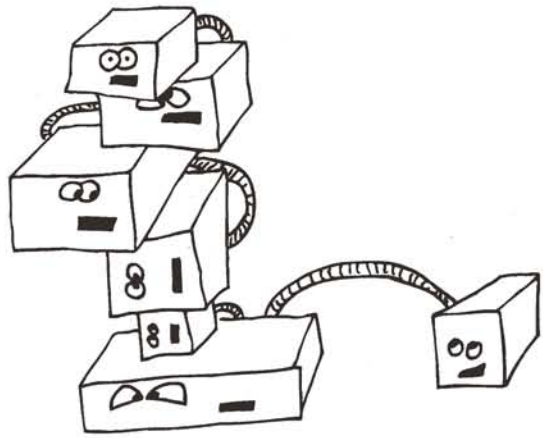
d）找出有重复登录名的所有项；

e）找出没有作废日期的所有项（需要**/etc/shadow** 文件）。

www.TopSage.com

大家网
TopSage.com

第 7 章　增加硬盘



真是令人难以置信，过去几十年来，虽然芯片、网络和软件技术都有长足的进步，我们却还在沿用 40 年前流行的长期数据存储技术。虽然存储密度已经提高了几个数量级（价格也降低了几个数量级），但是基本的原理却保持不变。

遗憾的是，硬盘空间的新用途还在不断涌现，特别是随着 Internet 的普及更是如此。在许多情况下，硬盘存储空间仍然不够用。MP3、视频流以及其他多媒体内容让系统管理员应接不暇。正确的硬盘管理仍然和以前一样重要。

大多数系统通过称为 SCSI（Small Computer System Interface，小型计算机系统接口，发音和“scuzzy”相似）的标准外设总线来连接硬盘。桌面和膝上 PC 支持另一种称为 IDE（Integrated Drive Electronics，集成驱动电路设备）的接口。我们通过讨论 SCSI 和 IDE 标准以及当今硬盘的结构来开始这一章的内容。然后我们将讨论硬盘格式化和分区的大体机制以及文件系统初始化的过程。最后，我们讨论 RAID 和卷管理器这类高级特性。

7.1　硬盘接口

现如今虽然刚刚出现了几种新技术，但只有少数几种接口标准得到了广泛的应用。选择和系统接口相匹配的硬盘驱动器非常重要。如果系统支持几种不同的接口，那么您就应该选择最能满足自己在速度、冗余性、移动性以及价格方面要求的一种。

- PATA（也叫做 IDE），是作为 PC 的一种简单而廉价的接口而开发的。因为它把硬件控制器和硬盘片本身放置在同一个盒子内，并且在计算机和硬盘之间使用一种相对高层的协议进行通信，所以最初它被称为“集成驱动器电路”。这就是所有现代硬盘的标准体系结构，但其名称沿用了下来。IDE 硬盘的速度中等、容量大，并且非常便宜。请参考下一节的内容了解关于 IDE 的更多信息。

www.TopSage.com

- 串行 ATA（SATA）是传统 IDE 的后续技术。除了支持更快的传输速率之外，SATA 通过清楚的连线和更长的最大电缆长度让连接更简单。SATA 本身支持热插拔和命令队列功能，这两种功能最终让 IDE 在服务器领域可以成为 SCSI 的一种替代选择。
- SCSI 虽然不如以前那么流行了，但它仍然是得到最广泛支持的硬盘接口之一。它有几种不同的形式，所有形式都支持在一条总线上安装多个硬盘，并且支持不同的速率和多种通信类型。SCSI 将在 7.1.3 节中更详细地介绍。
- 光纤通道（Fibre Channel）是在企业环境中正得到广泛应用的串行接口，因为它的带宽高，一次能够连接大量的设备。光纤通道设备通过光纤或者双铜轴电缆连接在一起。现在其速度能够达到 100 MB/s 以上。常见的拓扑包括 FC-AL（Fibre Channel Arbitrated Loop，光纤通道仲裁环路）的环形结构和使用光纤通道开关的交叉开关结构。光纤通道可以使用几种不同的协议，其中包括 SCSI 甚至 IP。光纤通道设备名可以由称为 World Wide Name 的硬连线 ID 号进行识别，它和以太网 MAC 地址类似。
- 对于连接键盘和鼠标这样的设备来说，USB（Universal Serial Bus，通用串行总线）已经开始变得流行起来，但它的当前版本也有足够的带宽来支持像硬盘和 CD-ROM 驱动器这样的设备。USB 在个人电脑上使用得非常普遍，它能让您很容易地在系统之间移动硬盘。

迄今为止，SCSI 和 IDE 在硬盘驱动器的领域内占据统治地位的角色。我们只详细讨论这两种接口。

7.1.1 PATA 接口

PATA（Parallel Advanced Technology Attachment，并行的先进技术附件规范），也称为 IDE，它的设计简单而且价格不贵。在个人电脑或者价格便宜的工作站中最容易找到它。IDE 在 20 世纪 80 年代末期开始流行。那之后不久，就又研发出 ATA-2 来满足用户和硬件驱动器厂商不断增长的需求。

ATA-2 增加了更快的 PIO（Programmed I/O，编程 I/O）和 DMA（Direct Memory Access，直接内存访问）模式，并且扩充了总线的即插即用功能。它还增加了一种名为 LBA（Logical Block Addressing，逻辑块寻址）的功能，此功能（和增强的 PC BIOS 组合起来）解决了 BIOS 无法访问硬盘上超过前 1024 柱面的区域这一问题。以前，这个限制把硬盘空间限定在 504MB 内。谁会想到硬盘会变得那样大呢！

因为 BIOS 管理了部分的引导过程，所以过去必须在前 1024 个柱面中创建一个小的可引导分区，以确保内核能够被老的 BIOS 载入。一旦内核载入并开始运行，那么就不需要 BIOS 了，可以访问硬盘的其余部分。这个不高明的策略在如今的硬件中是没有必要的，因为 LBA 摆脱了柱面-磁头-扇区（CHS）的寻址方式，而支持一种线性的寻址方案。

ATA-3 增加了更高的可靠性、更加精密的电源管理和自监控能力。Ultra-ATA 试图在 ATA-3 和 ATA-4 之间的差别上架起一座桥梁，它增加了名为 Ultra DMA/33 和 Ultra DMA/66 的高性能模式，它们分别将总线带宽从16MB/s扩展到了33MB/s和66MB/s。ATA-4也是一种需求很大的尝试，它将ATA-3和 ATAPI（ATA Packet Interface，ATA 包接口）合并在了一起，ATAPI 是能让 CD-ROM 和磁带机在一个 IDE 总线上工作的协议。

这一系列标准中最新的成员ATA-5和ATA-6增强了管理性能和出错处理功能，两者都会改善Linux这样的多用户环境的性能。ATA-7 预计是对并行 ATA 的最后升级。它支持的数据传输速率高达133MB/s。

IDE 硬盘几乎总是内置使用的（除非出于测试目的而考虑将硬盘挂在计算机外部）。ATA-2 总线的最大长度仅仅是 18 英寸，这就使总线甚至难以连到系统最上面的硬盘槽的位置。除了电缆的长度很短之外，一条 IDE 总线只能容纳两个设备。为了弥补这些不足，大多数制造商在他们的主板上提供了不止一条 IDE 总线（一般有两条，称为 primary 和 secondary）。

较老的 IDE 设备是以一种有连接的方式进行访问的，这就意味着一次只有一个设备是活动的。因

www.TopSage.com

此，如果将老式设备分散在多条总线上的话，性能将是最好的。扩展的 IDE 标准通过在一个接口上模拟两个 IDE 接口的方法绕过了这一限制，能让两个设备同时活动。当然，两个设备不能同时在电缆上发送数据，但是 SCSI 也是这样的。

IDE 连接器是一个 40 针的接头，它用带状电缆将硬盘连接到接口卡上。诸如 Ultra DMA/66 这样比较新一些的 IDE 标准使用一种不同的电缆，它能提供更多的接地针，因此减少了电路噪声。如果电缆或者硬盘没有标记，那么就应该确保硬盘上的针 1 和接口卡上的针 1 相连接。在连接器一边的针 1 通常会以一个很小的“1”来标记出。如果没有进行标记，那么规则就是，针 1 通常是那个离电源连接器最近的针。带状电缆上的针 1 通常用红色标记出来。如果在电缆的边缘部分没有红色条纹，那么就应该确保电缆的方向正确，使得针 1 和针 1 连接在一起，并且还应该清楚地标记出其位置以供下次使用。

如果在一条 IDE 总线上有不止一个设备，那么就必须将其中的一个设备指定为主设备，将另一个设备指定为从设备。现代的硬盘上有一个“cable select（电缆选择）”跳线设置（通常是默认），让设备自己选择是作为主设备还是从设备运行。作为主设备运行并不会有任何性能上的提升。有些比较老的 IDE 硬盘不喜欢作为从设备使用，所以如果在让一种配置工作的过程中遇到了困难，那么不妨试着颠倒一下它们之间的角色，让另一个设备作为从设备。如果仍然不能够正常工作，那么让每个设备都成为各自 IDE 总线上的主设备，这样情况可能会好一些。

考虑 IDE 硬件的时候，请记住以下这些要点。

- 新的 IDE 硬盘可以在老的板卡上工作，老的 IDE 硬盘可以在新的板卡上工作。自然，只有两个设备所共有的那些功能才能得到支持。
- 电缆长度非常短，只有 18 英寸，这就需要延长总线长度才能向总线中添加额外的设备。如果遇到了一些不确定出现的问题，那么应该检查一下电缆长度。自己定制的电缆可能会引发问题。
- IDE 设备新的电缆设计技术采用了缠绕在一起的打线方式，而不是更常见的带状电缆。新的电缆让机箱内部更整洁，而且改善了空气流通。

7.1.2 SATA 接口

随着 PATA 硬盘的传输速率不断提高（特别是随着 ATA-7 的出现），这项标准的缺点开始变得突出起来。电磁干扰和其他电气方面的问题开始导致在高速下出现可靠性问题。于是出现了 SATA（串行 ATA，Serial ATA）来解决这些问题。虽然 SATA 的概念在 2000 年左右的时候就有了，但是直到最近它才出现在消费级的工作站产品上。

SATA 缓解了 PATA 的许多突出问题。虽然 SATA 硬盘起初 150MB/s 的速度只比 ATA-7 稍微快一点儿，但是 SATA-2 在 2008 年前将最终支持高达 600MB/s 的传输速率[1]。其他显著的改进包括：更好的出错检查技术、能够热更换硬盘、自带命令队列机制、以及各种提高性能的措施。最后，SATA 不需要指定主从设备，因为一个通道只能连接一个硬盘。

SATA 克服了 PATA 电缆不能超过 18 英寸长的限制，还分别引入了电缆以及 7 针和 15 针连接插头的新标准[2]。这两种电缆比以前的带状电缆更灵活，用起来也更容易——为了在一条线上连接多个硬盘不用多弯曲和打折。

7.1.3 SCSI 接口

只要几个芯片集就能实现 SCSI 标准，所以有时候厂商就直接在主板上支持 SCSI。SCSI 定义了一

[1] 我们真的会有能以 600MB/s 的速率读写的硬盘吗？不会，但是即便目前，总线也不再是一个限制因素了。

[2] 这个说法是正确的，出于某种原因，电源线要比数据线更复杂一些。

www.TopSage.com

个通用的数据通道，所有类型的外围设备都可以使用这个通道。以前，数据通道由硬盘、磁带机、扫描仪和打印机使用，而现如今大多数外设都摈弃了 SCSI 而采用 USB。SCSI 标准并没有规定硬盘的结构或者布局如何，它只是规定了硬盘和其他设备进行通信的方式。

SCSI 标准已经经历了数次修正，目前的版本是 SCSI-3。SCSI-1 是在 1986 年作为基于 SASI（Shugart Associates System Interface）的一种 ANSI 标准而开发的，SASI 是一种需要商业购买才能采用的系统总线。SCSI-2 在 1990 年开发出来。它向后兼容 SCSI-1，但增加了几种提高性能的特性。这些特性包括命令队列，该功能让设备重新排列 I/O 请求以优化吞吐量，以及分散聚集 I/O，该功能可以在不连续的内存区域中进行直接内存访问（DMA，Direct Memory Access）传输。

您可能会看到人们使用术语“fast”和“wide”形容 SCSI-2 设备，它们的意思是说总线的速度提高了一倍，或者是说同时传输的位数更多了，典型情况下是用 16 位或者 32 位而不是通常所用的 8 位[3]。wide（宽的）SCSI 链能够支持多达 16 个设备，而 narrow（窄的）SCSI 电路只能支持 8 个设备。快速性和宽广性是互相独立的功能，通常情况下它们可以一起使用来提高综合性能。

SCSI-3 实际上是一系列标准。它包括了对不同物理介质的规范，这些物理介质既包括传统的并行总线，也包括像光纤通道以及 IEEE1394（“火线”）这样的高速串行介质。它还规定了 SCSI 的命令集合，并且增强了对设备自动配置、多媒体应用和新型设备的支持。

尽管 SCSI-3 的规范从 1993 年起至今就一直在制定之中，但仍然没有完成。不过，它的很多功能早就已经出现在市场上了，通常是以“Ultra SCSI”的名称出现。SCSI-3 中包含了 SCSI-2 的功能，所以 SCSI-3 内已经具有了一定程度的向后兼容性。不过需要牢牢记住的是，将老设备安装在新总线上会降低整个总线的速度。它还会影响电缆的最大长度。

表 7.1 总结了不同的 SCSI 版本和它们相应的总线带宽以及电缆长度的联系。

单端的 Ultra 和 Wide Ultra SCSI 最大电缆长度取决于投入使用的设备的数量。对于 8 个设备来说，最长为 1.5m，如果只用 4 个设备，那么总线长度就扩展到 3m。Wide Ultra SCSI 只能以差动模式支持 16 个设备。

表 7.1 SCSI 的发展

版 本	频 率	宽 度	速 度	长 度	差动长度
SCSI-1	5 MHz	8 bit	5 MB/s	6m	25m
SCSI-2	5 MHz	8 bit	5 MB/s	6m	25m
Fast SCSI-2	10 MHz	8 bit	10 MB/s	3m	25m
Fast/wide SCSI-2	10 MHz	16 bit	20 MB/s	3m	25m
Ultra SCSI	20 MHz	8 bit	20 MB/s	1.5m[a]	25m
Wide Ultra SCSI[b]	20 MHz	16 bit	40 MB/s	1.5m[a]	25m
Wide Ultra2 SCSI[b]	40 MHz	16 bit	80 MB/s	-[c]	25m(HVD)[d] 12m (LVD)
Wide Ultra3 SCSI[e]	80 MHz	16 bit	160 MB/s	-[c]	12m (LVD)

a. 有所不同。请参看下面的注释。
b. Wide Ultra SCSI 和 Wide Ultra2 SCSI 有时候分别被称为 Fast-20 Wide SCSI 和 Fast-40 Wide SCSI。
c. 这些版本的 SCSI 只使用差动信令。
d. HVD 是 High Voltage Differential（高电压差动信号），LVD 是 Low Voltage Differential（低电压差动信号）。HVD 用于早期的 SCSI 版本中，并不是在 Ultra2 SCSI 标准之上定义的。
e. Wide Ultra3 SCSI 有时候被称为 Ultra-160。类似的还有分别将速率翻倍和三倍的标准 Ultr-320 和 Ultra-640。大多数新硬盘是 Ultra-320 的。

SCSI 设备可以使用很多类型的连接器。它们根据所用的 SCSI 版本和连接类型的不同而异：内置

[3] 32 位 SCSI 总线不是太常见。有些可能需要多条电缆，分别称之为 A 电缆和 B 电缆。

www.TopSage.com

式或者外置式。narrow（窄的）SCSI 设备有 50 个针，wide（宽的）SCSI 有 68 个针。内部设备通常接受一个 50 针接头或者连接在带状电缆上的 68 针的针式迷你型微连接器。外部设备通常使用高密度的 50 针或者 68 针迷你型微连接器和计算机进行连接。

一种有趣的 SCSI 变体对于热插拔硬盘阵列尤其有用，这就是 SCA（Single Connector Attachment，单连接器附件）插头。它是一个包含有总线连接、电源和 SCSI 配置的 80 针连接器，因而可以让一个单独的连接器满足硬盘的所有需要。

图 7.1 中显示了最常用的连接器的图片。每个连接器都是从前面显示的，就好像您正要把它插入自己的前额一样。

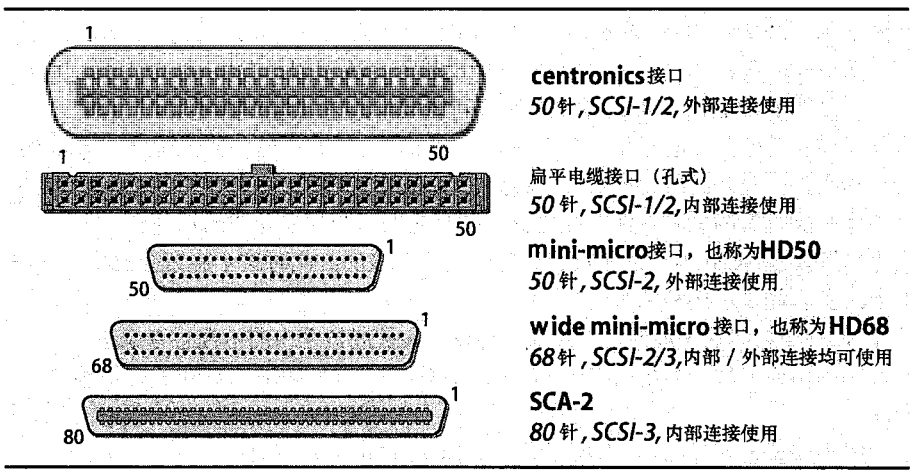


图 7.1 通用的 SCSI 连接器（前视图，除了已注明的之外都是针式的连接器）

SCSI 总线使用一个菊花链的配置，所以多数外部设备具有两个 SCSI 端口。这两个端口是相同的，并且可以互换，所以任何一个都可以是输入端口。由于一些原因，扫描仪厂商认为他们自己可以不受这些固有规则的限制，所以有时候只提供一个 SCSI 端口。如果没有在内部加以终结，那么这些设备就需要一种特殊类型的终结器。

内部 SCSI 设备经常被连接到一条带状电缆上。因为连接器能够被压接到带状电缆的中间，所以实际的 SCSI 设备上只需要一个端口。当使用带状电缆时，一定要确保 SCSI 总线上的针 1 是和硬件驱动器上的针 1 相连接（针 1 通常用红色条纹标记）。

SCSI 总线的每一端都必须有一个终结电阻（“终结器”）。当信号到达总线末端时，这些电阻会吸收这些信号，以防噪声返回到总线中去。终结器有几种形式，从可以插到常规端口上的小型外部插头到安装在设备电路板上的一系列很小的排电阻，都有可能。大多数现代设备是自动终结的。

总线的一端通常在主机内部终结，不是在 SCSI 控制器上终结，就是在一个内部 SCSI 驱动器上终结。总线的另一端通常会在一个外部设备上终结，如果没有任何外部设备的话，就会在 SCSI 控制器上终结。如果您的 SCSI 总线出现了似乎是随机发生的硬件问题，那么就应该首先检查总线两端是否正确终结了。不正确的终结是最常见的 SCSI 配置错误，它产生的错误可能是非常不明显并且是时断时续的。

SCSI 地址本质上可以是任意的。从技术上看，它决定了总线上设备的优先权，但在实际使用中，精确的优先权并不会有很大的差别。一些系统将目标号最低的硬盘作为默认的引导盘，还有一些系统要求引导盘为目标 0。

如果运气好的话，碰到的设备会有一个外部小旋钮，使用它可以设置目标号。其他设置目标号的常见方法是 DIP 开关和跳线。如果不能明确在设备上如何设置目标号，那么就应该参考硬件手册。现

www.TopSage.com

在大多数硬件说明都可以在制造商的网页上找到，可以尝试着用一个硬盘进行大量的试误测试来进行设置。

SCSI 标准支持一种名为“逻辑单元号”（LUN，Logical Unit Number）的子地址。每个目标内部都有几个逻辑单元。例如有几个硬盘但只有一个 SCSI 控制器的硬盘阵列。然而，逻辑单元很少在实际生活中用到。当您听到“SCSI 单元号”的时候，应该假定它就是正在讨论的目标号，除非事实证明是其他的东西。如果 SCSI 设备只包含一个逻辑单元，那么 LUN 经常默认设置为 0。

通常情况下，SCSI 总线很容易配置，但有一些事情也可能会出错。

- 很多工作站都具有内部 SCSI 设备。在重新启动添加新设备之前应该检查一下目前所有设备的清单。
- 在添加了一个新的 SCSI 设备之后，检查 OS 重新启动时所发现的设备清单，从而确认所期望有的所有设备都在那里。大多数 SCSI 驱动器不会检测到具有相同 SCSI 地址的多个设备——这是一种非法的配置。SCSI 地址的冲突可能会导致非常奇怪的现象。
- 有些扩展盒（包含有电源和一个或者多个 SCSI 设备的盒子）在盒子内部终结总线。如果设备是附加在扩展盒之后的，那么您就会遇到 SCSI 链上所有设备都不可靠的问题。无论什么时候，都要仔细检查您的确具有两个终结器，并且它们两个都在总线的端部。
- 用于设置设备 SCSI 地址的小旋钮有时候是倒着接的。此时这个旋钮会改变 SCSI 地址，但是不会改变已经显示出来的数值。
- 计算 SCSI-2 总线的长度时，应该确保将设备和扩展盒之内的电缆也考虑进去。这些电缆可能非常长。还要记住，如果将老的 SCSI 设备添加到新的 SCSI 总线中去，电缆的最大长度可能会减少。
- 永远都不要忘记 SCSI 控制器使用了一个 SCSI 地址。

7.1.4 SCSI 和 IDE 的比较

在本书的前面几版里，SCSI 在服务器上应用一直都处于明显的优势。不过，随着 SATA 硬盘的出现，SCSI 没有以前那么值得投资了。SATA 硬盘能够和几乎每一类 SCSI 硬盘相媲美。与此同时，SATA 硬盘又非常便宜，使用更为广泛。

SCSI 硬盘的优势之一是在控制器上集成了一个处理器，能够把系统的 CPU 解放出来去处理其他事情。在繁忙的系统上，这就意味着获得了很大的性能提升。当然，要由用户自己确定花这么多钱是否值得。

在有些情况下 SCSI 是可取的，甚至是必须采用的。

- 如果必须具有可能的最好性能，那么应选择 SCSI。硬盘制造商们用 IDE/SCSI 这样的划分来满足硬盘驱动器市场的需求。有些 IDE 驱动器可能在吞吐速率的峰值上超过了 SCSI，但是 SCSI 几乎一定会提供更好的持续吞吐速率。
- 服务器和多用户系统需要使用 SCSI。SCSI 协议在以高效方式管理多个并发请求的能力方面是无与伦比的。在一个繁忙的系统中，您会看到系统性能得到了提高，这是具体的，能够测量到的。
- 如果需要连接很多设备，那么 SCSI 又领先一筹。SCSI 设备可以和其他设备很好地协作，而 IDE 设备会在总线上互相干扰和冲突。

7.2 硬盘的几何结构

图 7.2 显示了硬盘的几何结构以及通常用于称呼硬盘各个部分的术语。之所以提供这些信息，主

www.TopSage.com

要是为了提高读者知识的全面性。现代的硬盘驱动器仍然以同样的基本结构为基础，但软件却再也不知道（或者是不必知道）关于驱动器的物理结构的很多信息。

典型的硬盘驱动器由镀上磁性薄膜的旋转盘片组成。数据是通过改变盘片表面磁性微粒方向的一个小磁头来进行读取和写入的。数据盘片是完全密封的，这样任何灰尘或者杂质都进不去。这项功能使固定的硬盘比可移动的介质更加可靠。

在计算机硬件的很早时期，硬盘驱动器通常只有一个盘片。存储能力的增加是通过盘片直径的增加来做到的。在我们的一个用户区的墙上有一个很古老的硬盘，其直径超过 4 英尺，保存了大约 280KB 的数据。

现在，硬盘通常都有几个互相叠在一起的小盘片，而不是只有一个单独的大盘片。盘片的两面都用来保存数据，尽管有一个硬盘的一面通常用来保存定位信息，而不能用来存储。单盘密度目前能够高达约 130GB。

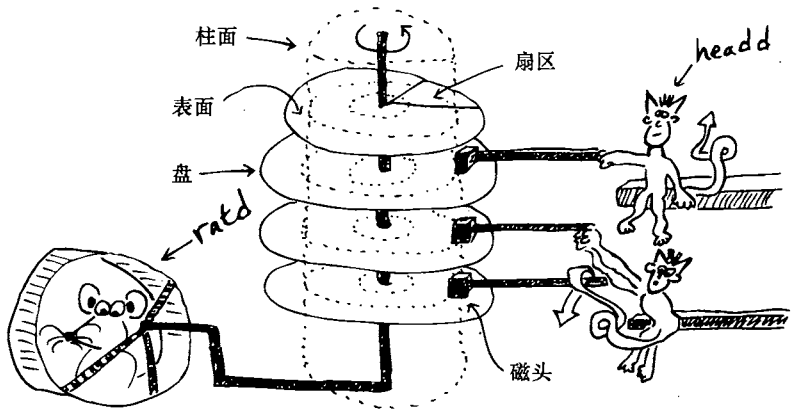


图 7.2 硬盘的几何组织

盘片以恒定的速度旋转。它们通过很小的滑动磁头进行读取和写入，这些小磁头就像唱针在唱片机上一样前后移动。磁头漂浮在距离盘片表面非常近的位置，但并没有实际接触盘片。磁头和旋转盘片之间的距离可以和 F-16 喷气式战斗机以全速在距离地面 10 英尺的高度处飞行相提并论。如果磁头接触到了盘片，那么这就被称作磁头碰撞，这种情况是非常具有破坏性的。

硬盘的旋转速度随着时间推移已经得到了大幅增加。老式硬盘只能以 3600RPM（转 / 分）或者 5400RPM 的速度运转。现在 7200RPM 是主流市场的标准，而且 10000RPM 和 15000RPM 的驱动器正在高端市场上流行起来。更高的旋转速度减小了等待时间、增加了数据传输的带宽，但这可能会引入一些由于产生的热量增加而滋生的散热问题。如果您打算购买一个高端驱动器，那么一定要确保有充分的空气流通性。

每个表面至少需要一个磁头。早期驱动器上的磁头必须移动很大的距离，但现在这种小的堆叠盘片的几何结构效率更高一些。硬盘直径仍然在继续减小，从 20 年前的 14 英寸的标准减小到了 10 年前的 5.25 英寸的标准，到现在只有 3.5 英寸，甚至更小一些。

将磁头移动到正确位置读取特定的一段数据，这称作寻道。当硬盘在磁头下面旋转时磁头所能达到的每个位置称为磁道。磁道还可以进一步划分为扇区，通常情况下它有 512 字节长。

在不同的盘片上距离旋转轴相同距离的一系列磁道称为一个柱面。如果所有磁头都一起移动（这在占市场份额最大的驱动器中是很典型的），那么不需要任何额外的移动，就能够读取存储在一个单独柱面上的数据。尽管磁头的移动相当快，但它们仍然比硬盘的旋转要慢得多。因此，任何不需要磁头寻找新位置的硬盘访问都会更快一些。

www.TopSage.com

7.3 Linux 文件系统

随着 Linux 从各种不同的操作系统吸收了许多特性，Linux 上的文件系统一直在快速演变。内核的 VFS（Virtual File System，虚拟文件系统）层对于 PC 来说特别方便，因为它提供了安装“原始”文件系统（比如臭名昭著的 Windows FAT 文件系统）所需要的框架。

7.3.1 ext2fs 和 ext3fs

Second Extended File System，也常称为 ext2fs，在过去很长时间里一直是主流的 Linux 文件系统。它主要是由 Rémy Card、Theodore Ts'o 和 Stephen Tweedie 设计和实现的。虽然 ext2 的代码是专为 Linux 编写的，但是它采用了许多源自于 BSD FFS 文件系统的概念，后者由 Kirk McKusick 和他的小组在 1984 年设计和实现。

ext3fs（Third Extended File System）是对 ext2fs 的一种非常知名的扩充，它最初由 Stephen Tweedie 开发，现在是大多数 Linux 发行版本默认的文件系统。ext3fs 向现有的 ext2fs 代码加入了日志功能，从概念上说很简单的修改却大大增加了可靠性。更有意思的是，甚至不必改变 ext2fs 的基础结构，就可以实现 ext3fs 扩展。实际上，您仍然可以把一个 ext3fs 文件系统当作 ext2fs 文件系统来安装——它只是不会启动日志功能而已。

用 **tune2fs** 命令就能把一个 ext2fs 文件系统转为 ext3fs 文件系统。例如，如果在**/dev/hda4** 上有个 ext2fs 文件系统，那么可以用下面的命令进行转换：

```
# tune2fs -j /dev/hda4
```

接着要修改**/etc/fstab** 中对应于该分区的配置项，使之读取 `ext3` 而不是 `ext2`（参见 7.4.5 节了解 **fstab** 文件的更多知识）。

ext3fs 文件系统在硬盘上专辟了一个区域来保存日志文件。在发生文件系统操作的时候，所要求的修改首先写入日志文件。在完成日志更新之后，写入一条“提交记录（commit record）”标记日志项的结束。只有这样以后，才对正规的文件系统作修改。如果发生了崩溃，可以用日志记录重构出完全相同的文件系统。

日志机制将执行文件系统一致性检查所需的时间（参见 7.6 节介绍的 **fsck** 命令）减小到每个文件系统大约 1 秒。除了某种硬件故障之外，ext3fs 的状态几乎能立即评估和恢复。

本章只详细介绍 ext3fs 这一种文件系统。

7.3.2 ReiserFS

Hans Reiser 开发的 ReiserFS 是另一种在 Linux 上新出现的文件系统，SUSE 把它作为自己默认的文件系统。和 ext3fs 一样，ReiserFS 也是一种日志文件系统，因此能够维护文件系统的一致性，不必担心诸如系统崩溃、意外重启这样的异常事件（这在膝上或者桌上工作站环境里是很常见的）。2004 年 8 月发布的 ReiserFS 第 4 版包含在 Ubuntu 的默认安装中。Fedora 和 SUSE 迄今仍然在使用较早的版本。

除了日志功能之外，Reiser4 还会提供一种模块化的文件系统接口，通过这个接口，应用软件开发人员和系统管理员可以在非常细的粒度上指定应该怎样处理（和保护）文件。这项功能有可能在特殊

www.TopSage.com

环境里增强文件的安全性。ReiserFS 由 DARPA[4]提供资助，它是声称为达到军用级别的安全性而设计的唯一开放源代码的文件系统。

Reiser4 的新算法比以前的算法空间效率更高。别的文件系统（以及 ReiserFS 的较早版本）使用平衡树算法来分配地址空间块。虽然这种尝试性的方法经证实很可靠，但是它一般会在速度和磁盘利用率之间进行折中。Reiser4 的“dancing tree”算法不要求系统管理员在速度和磁盘利用率两方面进行选择。

7.3.3 XFS 和 JFS

在这类“现代、主流的 Linux 文件系统”中还有两个竞争者，它们就是 SGI 的 XFS 和 IBM 的 JFS。目前的内核和发行版本都支持它们，它们都是高性能的日志文件系统，而且有自己的支持者，但它们似乎都没有 ext3fs 或者 ReiserFS 使用广泛。

很难在任何程度上解释清楚为什么会这样，但是根本的问题似乎在于，这两种文件系统同 ext3fs 和 ReiserFS 没有太大不同。说实话，它们的实现细节不一样，每种文件系统都有一定的优势领域（例如，XFS 的碎片整理更好，管理范围更广）。然而这两种主要竞争者的性能却类似，能让其中任何一个文件系统体现出优势的应用软件都很少。

如果想要详细评估这两个文件系统，可以在网上找到更多资料。对于 JFS 来说，可以看看 jfs.sourceforge.net，而 XFS 可以访问 oss.sgi.com/projects/xfs。

7.4 硬盘安装过程概述

添加新硬盘的过程包括以下这些步骤：

- 将硬盘连接到计算机上；
- 如果必要的话（通常不需要），格式化硬盘；
- 为硬盘添加卷标以及对硬盘进行分区；
- 建立逻辑卷；
- 在硬盘分区中创建文件系统；
- 设置自动安装机制；
- 在交换分区上设置交换机制。

下面各小节将详细介绍这些基本步骤，从 7.7 节开始，我们将以一个硬盘驱动器为例，从头到尾完整介绍安装过程。

7.4.1 连接硬盘

将硬盘添加到系统中的方式取决于硬盘所用的接口。剩下的工作就是安装托架和连上电缆。如果是 IDE 硬盘，那么就应该试着把系统配置为在每条总线上只安装一个 IDE 硬盘。应该仔细检查电缆的方向和每个硬盘上的主/从设置。如果是 SCSI 硬盘，那么就应该仔细检查，确保已经正确终结了 SCSI 总线的两端，电缆长度要比所用 SCSI 的最大电缆长度短，以及新 SCSI 设备的目标号不应该与控制器以及总线上的其他设备发生冲突。

在可以访问一个新硬盘之前，需要在**/dev** 下有指向它的文件。Linux 自动为所有可能的 SCSI 设备

[4] 国防部高级研究计划署，与资助创立 Internet 的是同一个单位。

www.TopSage.com

创建了文件。参考 5.4.3 节了解设备文件的全面知识。

向硬盘随机执行写操作有可能在几秒之内毁掉一个文件系统，所以应该相当严格地设置硬盘文件的访问权限。假如让组的属主（“operator”或者“disk”）有读权限，这样的设定可以让操作员无需超级用户特权就能执行转储操作（**dump**），但又能避免普通用户读原始设备。

7.4.2 格式化硬盘

厂商经常用没有经过格式化的字节数来标明硬盘的容量。硬盘容量的大约 10%一般会用于标记硬盘的表面，以便让系统硬件和软件可以找到已经写入数据的地方。在买硬盘的时候，应该始终用格式化之后的大小来考虑硬盘的容量和比较价格。

硬盘的容量用的兆字节是百万字节数，这和内存标称不同，内存的兆实际上是 2^{20} 或者 1 048 576 字节。这种差别大约有 5%[5]。所以，在估计和比较硬盘容量时，一定要检查它们的单位。

格式化硬盘的过程就是在盘片上写入地址信息和时间标志来划分每一个扇区。它还能标出“坏块”，也就是介质上的一些会导致不能在硬盘某个区域内进行可靠的读或者写操作的瑕疵。现代硬盘具有内置的坏块管理功能，所以无论是您还是驱动程序都不必担心这个问题[6]。所有的硬盘都要事先格式化好，制造工厂对硬盘的格式化通常比您所能进行的任何格式化都要精确得多。如果没有必要的话，最好应该避免进行低级格式化。如果遇到了硬盘读写错误，那么首先应该检查电缆、终结以及地址方面的问题，这些问题也能够引起和硬盘坏区相类似的症状。如果完成了这个过程之后您仍然确信硬盘坏了，那么使用一个新硬盘来替换该盘会比为完成硬盘的格式化而等待很长时间更好一些。

IDE 硬盘在通常情况下是不能在制造工厂之外进行格式化的。但您可以从制造商那里获得专用的格式化软件，通常情况下在 Windows 中使用。应该确保所得到的软件与自己打算格式化的硬盘相匹配，并且要仔细遵循制造商的指导操作。

SCSI 硬盘能够响应您从主计算机中发出的命令，自己完成格式化。发送该命令的过程随系统不同而异。在 PC 上，通常可以从 SCSI 控制器的 BIOS 中发送该命令。

借助各种不同的工具，您可以通过在硬盘上随机写入然后进行读取来检验硬盘的完整性。这个过程非常花时间，所以，除非怀疑硬盘坏了而且不能简单地替换它（或者在乎时间），可以跳过这个过程。除此之外，可以让这个测试过程在夜间运行。不要担心过多的使用或者过分的检测会对硬盘造成磨损。因为硬盘的设计可以经受持续不断的操作。

7.4.3 标记硬盘和硬盘分区

在对硬盘进行格式化并标记了坏扇区之后，就必须将它划分成分区这样的大块区域。硬盘分区能够把硬盘当作一组独立的数据区域，而不是一个巨大的数据块集合。硬盘分区中还有引导块和分区表这样的结构，它们本身对于高层软件（比如说，文件系统）而言是不可见的。只有设备驱动程序知道整个硬盘的布局结构，其他软件都是在整理过的抽象分区上进行工作的。

分区可以使备份更容易、可以防止用户窃取其他用户的硬盘空间、提高系统性能以及限制来自失控程序的潜在损害。分区表保存在硬盘上一个称为卷标的记录中。卷标通常占据硬盘上最开始的几个数据块。它的具体内容各不相同，但它一般都包括了保证系统启动的足够信息。

在概念上，硬盘上的各个分区都不同，并且彼此之间互相独立。然而，**/dev/sda** 是硬盘的块设备文件，它实质上是整个硬盘的映像。用户级的命令可以通过这个设备文件来“直接”访问硬盘。例如，

[5] 当然，“mega”的前缀实际上就是指“million”，所以那种用法并不是完全没有理由的。

[6] 然而，硬盘格式化之后出现的任何坏区都是不能“处理的”。它们将通过读写错误以及丢失数据的形式表现出来。

www.TopSage.com

用户级的进程能够通过使用 **dd** 命令来写硬盘的卷标，或者将它的内容复制到一个备份硬盘中。当然，必须很小心地使用这个特殊的文件，因为它会一下子就破坏硬盘上的每个分区。

有些系统甚至在这条危险的路上走得更远，这样的系统允许定义多个重叠的分区组。例如，分区 1、2 和 3 可以按一种方式来划分硬盘，而分区 4 和 5 又可以使用另外一种方式划分硬盘。您应该使用一组自相容的分区，然后忽略其他所有的分区组。在实际使用中，这样的重叠分区会引起操作者的错误，它是数据不时发生损坏的常见原因。

现在的系统倾向于比以前的系统使用更少的分区，但在大多数系统里必须至少有下面两种分区。

- 根分区：让系统进入单用户模式所需要的所有东西都保存在这里。这个分区的一个副本通常保存在另外一个硬盘中，以供应急之用。
- 交换分区：当没有足够的物理内存来保存信息时，交换分区就能存储虚拟内存中的内容。每个系统都至少应该有一个交换分区。

划分硬盘分区的最好方式如何选择则会有不同，下面是一些提示。

- 如果有多个硬盘，那么在其中的一个硬盘上复制一份根文件系统，检验一下是否可以从该硬盘进行启动。
- 在给机器添加内存时，还应该增加交换分区的空间。根据满足系统上工作量的需要来分配交换空间的大小。参见 25.3.2 节了解更多有关虚拟内存的知识。
- 在几个硬盘上分摊交换分区可以增强系统的性能。这项技术也能对文件系统起作用，也就是说，把那些很繁忙的文件系统放置在不同的硬盘上。请参看 25.3.4 节以获取关于这个话题的说明。
- 如果想要备份一个分区，那么不要让这个分区比备份设备的容量还要大。请参考 9.1.6 节。
- 把变化较快的信息集中在几个分区上，并经常备份这些分区。
- 把**/tmp** 放在一个独立的文件系统上，把临时文件大小限制在有限的范围，能够省去备份操作。
- 因为日志文件保存在**/var** 中，所以将**/var** 作为一个独立的硬盘分区是个好主意。让**/var** 作为一个很小的根分区中的一部分，很容易使其填满根区，导致机器进入暂停状态。
- 把**/home** 定为一个独立的分区比较有用。即使根分区出问题或者损坏了，用户数据也有很大机会没有受到影响。相反，甚至在被诱导的用户 shell 脚本填满了**/home** 之后，系统仍然能够继续运行。

7.4.4 在硬盘分区内创建文件系统

即便一个硬盘已经在概念上被分成了若干个分区，它仍旧没有为保存文件做好准备。在硬盘为使用准备就绪之前，文件系统需要加一点儿自己的开销。

要在一个硬盘分区里安装一个文件系统，需使用命令 mke2fs。要在一个硬盘分区上安装一个 ext3fs 文件系统，可以使用 **mke2fs -j**（没有 **mke3fs** 这样的命令，**-j** 选项告诉 **mke2fs** 创建 ext3fs 的日志）命令。除非使用的系统有特殊之处，否则只要用 **mke2fs -j** 命令加上分区名就应该能建立文件系统。使用 **mke2fs** 要小心，用错了会毁坏系统，即便有可能恢复数据，也极其困难。

ext3fs 文件系统包括 5 个组成部分：

- 一组索引结点（inode）存储单元；
- 一组分散的“超级块（superblock）”；
- 一份文件系统中磁盘块的映射表；
- 一份磁盘块使用情况的汇总；
- 一组数据块。

日志就仿佛它是新文件系统根目录中的一个普通文件那样来进行分配，所以它实际上并不是一个

www.TopSage.com

特别的结构部分。

每个文件系统分区都被分成若干个块簇。诸如索引结点表这样的结构则在块簇间进行分配，于是，要一起访问的若干磁盘块可以在硬盘上彼此保存得很靠近。这种成簇机制减少了在访问同一个文件的磁盘块时，需要搜索整个硬盘的情况。

inode 是长度固定的表项，每个 inode 都保存着关于一个文件的信息。因为在初始建立文件系统的结构时就开辟出一块空间用于保存 inode，所以必须事先判断要创建多少个 inode。由于不可能准确预测将来有一天会需要多少个文件（inode），所以建立文件系统的命令使用一个经验公式，根据分区的大小以及文件的平均长度来猜测一个合适的数值。

在创建文件系统的时候可以把 inode 的数量往多或者往少调整：小文件多的文件系统（比如存放源代码库的文件系统）inode 多一些，保存几个大文件的文件系统（比如保存有数据库的文件系统）inode 少一些。

超级块是一种描述文件系统特性的记录。它包含的信息有：磁盘块的长度、inode 表的大小和位置、磁盘块的映射表和使用情况信息、块簇的大小、以及其他几个重要的文件系统参数。因为损坏超级块可能会丢失一些极其关键的信息，所以文件系统在硬盘上的分散位置（在每个块簇的起始处）保存有超级块的几个副本。

对于每个已经安装到操作系统中的文件系统来说，内核都会在内核里保存一份超级块的副本，也同时在硬盘上保留几份超级块的副本。系统调用 **sync** 把缓存在内存里的超级块冲洗（flush）到它们在硬盘上的永久位置上，瞬间就让文件系统保持了一致性。这种周期性的保存操作，能够把文件系统尚未更新超级块机器就发生崩溃时造成的损失减少到最小。**sync** 也会冲洗修改过的 inode 和被缓存的数据块。传统上每隔 30s 执行一次 **sync**，但是 ext3fs 却每隔 5s 执行一次。

文件系统的磁盘块映射表是它所包含的空闲块的一张表。在写新文件的时候，要研究这个映射表，设计一种有效的布局方案。块的使用情况汇总记录了有关已经投入使用的块的基本信息。

7.4.5 安装文件系统

进程可以使用文件系统之前必须先安装它。文件系统的安装点可以是任何目录，但是如果在某个目录上安装了一个文件系统，那么在它下面的文件和子目录都会访问不到了。参考 5.2 节了解有关安装文件系统的更多信息。

在安装一个新硬盘之后，应该手工安装新的文件系统来测试每样东西是否都能正确工作。例如：

```
# mount /dev/sda1 /mnt
```

命令会把设备文件**/dev/sda1** 所表示的分区里的文件系统安装到目录**/mnt**，这是用于临时安装文件系统的传统路径。如果是崭新的文件系统，那么它的内容应该看上去和下面的类似：

```
# ls /mnt
lost+found
```

当构建一个文件系统的时候，会自动创建一个 **lost+found** 目录。它在紧急情况下供 **fsck** 使用，不要删除它。**lost+found** 目录还预先分配了一些额外的空间，所以在一个不稳定的文件系统里，**fsck** 能够在那里保存“没有链接”的文件而不需要多分配目录项。

用 **df** 命令可以核实文件系统的大小。下面是一个例子：

```
$ df /home
Filesystem    1k-blocks     Used   Available   Use%   Mounted on
/dev/hda5       4128448   697968     3220768    18%   /home
```

www.TopSage.com

df 报告的单位是 1K 大小的块，或者可以使用**-h** 选项要求以好读的数值来显示结果（例如，2.4G）。

7.4.6 设置自动安装

用户一般希望配置系统在启动的时候自动安装本地的文件系统。**/etc/fstab** 文件包含与文件系统相对应的设备的清单。

包含前面文件系统的一个 **fstab** 文件可能看起来像下面这样：

```
LABEL=/         /              ext3      defaults          1 1
none            /dev/pts       devpts    gid=5,mode=620    0 0
/dev/hda5       /home          ext3      defaults          1 2
none            /proc          proc      defaults          0 0
none            /dev/shm       tmpfs     defaults          0 0
/dev/cdrom      /media/cdrom   iso9660   ro,noauto,owner   0 0
/dev/hda3       /usr           ext3      defaults          1 2
/dev/hda6       /var           ext3      defaults          1 2
/dev/hda2       swap           swap      defaults          0 0
```

每行有 6 个由空白隔开的字段。每行描述一个文件系统。传统上对齐各个字段是为了阅读方便，但并不要求对齐。

第 1 个字段给出了设备名，或者是 **e2label** 关联到设备上的标签（以 LABEL=的形式）。**fstab** 文件可以包括从远程系统上安装的文件系统，在这种情况下，第一个字段是 NFS 路径。记法 *server:/export* 指出的是名为 *server* 的机器上的*/export* 目录。实际不占用存储空间的文件系统（比如前面提到过的**/proc** 和**/dev/shm**）在这个字段都用 none 来占位。参考第 16 章了解有关 NFS 的更多信息。

第 2 个字段指定安装点，第 3 个字段指定文件系统的类型。标识本地文件系统所用的确切类型名称随系统的配置不同而不同。ext2 和 ext3 分别用于 ext2fs 和 ext3fs，reiserfs 用于 ReiserFS。

第 4 个字段列出了安装选项。关键字 defaults 代表 rw、suid、dev、exec、auto、nouser 和 async 这些选项的一种组合（参考 **mount** 命令的手册页了解每种选项的详细情况）。用户可能还想包含一个 acl 选项来打开访问控制列表，参见 5.6 节。

ext3 文件系统的日志管理模式用 data=*mode* 选项来设置，这里的 *mode* 可以是 ordered、writeback 或者 journal 之一。这个模式（*mode*）是运行方式的选择，而不是文件系统自身的属性，这也是它会出现在 **fstab** 文件中，却又不是 **mke2fs** 命令参数的原因。

第 5 个字段指出“转储频率”的值，从理论上说，可以由备份设备来使用，但通常不用。

- 默认的“ordered（规则）”模式保证文件系统始终保持一致性，因此文件决不会因为系统崩溃而受损。对于大多数环境来说，这是最好的选择。
- “writeback（回写）”模式在系统崩溃之后会产生受损文件，但在某些情况下运行速度更快。这实际上是大多数带有日志的文件系统的工作方式，它本质上并没有默认的 ext2 危险。
- “journal（日志）”模式使用的日志文件更大，这可能会降低系统重启时恢复文件系统的速度，但是在用于数据库应用时速度更快。

第 6 个字段指定 **fsck** 应该检查文件系统的次序。在这个字段中有相同值的文件系统会尽可能并发检查。在日志文件系统出现之前，**fsck** 是一个很花时间的进程，所以这个字段那时候非常重要，但是现在它没那么重要了。不要将位于同一硬盘上的两个非日志文件系统设为同样的值，否则将会导致磁头来回寻道，大大降低性能。只有位于不同硬盘上的文件系统才应该并行检查。**fsck** 将在 7.6 节介绍。

mount、**umount**、**swapon** 和 **fsck** 命令都要读取 **fstab** 文件，所以在这个文件里提供正确和完整的数据非常重要。当您在命令行上只给了分区名或者安装点的时候，**mount** 和 **umount** 命令就使用 **fstab** 文件来判断您想要做的事情是什么。例如，使用刚才给出的 **fstab** 文件，命令

www.TopSage.com

```
# mount /media/cdrom
```

等价于键入

```
# mount -t iso9660 -o ro,noauto,owner /dev/cdrom /media/cdrom
```

mount -a 命令会安装 **fstab** 文件里列出的所有的普通文件系统，它通常在引导时刻由启动脚本执行。**-t** 标志将操作限制在某种类型的文件系统。例如：

```
# mount -at ext3
```

会安装硬盘上所有的本地 ext3 文件系统。**mount** 命令顺序读取 **fstab** 文件，因此，在 **fstab** 文件中，在其他文件系统之下的文件系统必须在其父分区的后面出现。例如，如果**/var** 是一个单独的文件系统，那么用于**/var/log** 的配置行必须在**/var** 的配置行之后。

umount 命令卸载文件系统，它的语法和 **mount** 类似。当某个进程正在用一个文件系统作为它的当前目录或者该文件系统上面有打开的文件，就不能卸载这个文件系统。有办法绕过此项限制，参见 5.2 节。

7.4.7 启用交换分区

UNIX 很早就有的优点之一就是它能够实现虚拟内存。这一特性可以使操作系统假装认为机器拥有的内存比其实际内存多。如果进程想使用这种“多出来的”内存，那么就会把系统硬盘作为一种超慢的 RAM 投入使用。将内存中的内容像变戏法般地写入硬盘或者从硬盘中读出，这称为内存交换（swapping）或者页面调度（paging）。

为了使交换更加有效，一般要用原始分区（没有文件系统的分区）作为备份储存区。内核保留了它自己的从内存块到磁盘块的简化映射关系，而不是使用文件系统结构来跟踪交换区中的内容。还有可能把内存的内容交换到文件系统分区里的一个文件上去，但这样的配置比使用专门的分区要慢得多。

交换空间越多，进程能分配到的虚拟内存就越多。在几个硬盘之间分摊交换区时，就能获得最好的交换性能（在几条 SCSI 总线之间分摊性能会更好一些）。请参考 25.3.4 节以获取关于分摊交换区的更多信息。

您可以手工启用到一个特定设备的交换，但通常情况下需要使这项功能在系统引导时自动执行。交换区可以在 **fstab** 文件中列出来，和用来列出可安装文件系统的是同一个文件。一个交换区的配置项类似下面：

```
/dev/hda2       swap        swap        defaults        0   0
```

在启动过程中，系统会运行 **swapon** 命令来启用在 **fstab** 文件中列出的所有交换分区。

7.5 hdparm：设置 IDE 接口参数

hdparm 这个程序通过和 Linux 的 IDE 驱动程序打交道来获得和改变硬盘的参数。其中 hdparm 能够设置硬盘的供电模式、启用或者禁用 DMA、设置只读标志、以及打印详细的硬盘信息。通过调节这些硬盘参数中的一些，就能实现很大的性能提升。这个工具只适用于 IDE，在 SCSI 或者 USB 设备上不起作用。

调用 hdparm 命令的语法为：

www.TopSage.com

hdparm *[options] [device]*

不带参数运行 **hdparm** 命令会输出一些有意思的设置在当前的状态。

```
$ sudo /sbin/hdparm -d /dev/hdb

/dev/hdb:
    multcount    =  0 (off)
    IO_support   =  0 (default 16-bit)
    unmaskirq    =  0 (off)
    using_dma    =  0 (off)
    keepsettings =  0 (off)
    readahead    = 256 (on)
    geometry     = 36481/255/63, sectors = 300069052416, start = 0
```

从硬盘性能的角度来看，这些设置漏掉了许多必要的功能。

- DMA（直接内存访问，direct memory acess）被禁用了。所有的现代硬盘都能不经过 CPU 直接向内存发送数据。启用 DMA 几乎始终是一个好主意，但它偶尔会让设计不好的硬件发生故障。一定要先在非生产系统上进行测试。
- 32 位 I/O 支持被禁用了。没有 32 位 I/O，数据在 PCI 总线上以 16 位为单位进行传输，可能会让性能下降一半。
- 消除中断屏蔽的功能被关闭了。打开这项（有点不引人注意的）功能，可以让硬盘驱动程序在服务当前一次硬盘中断的同时消除对挂起中断的屏蔽。

hdparm 内有一项硬盘性能测试功能，它可以帮助评估这些设置造成的影响。我们的系统有一块比较老的 IDE 硬盘，在配置默认值的情况下，它的输出如下：

```
$ sudo /sbin/hdparm -Tt /dev/hdb

/dev/hdb:
    Timing cached reads:    228 MB in  2.41 seconds =  94.70 MB/sec
    Timing buffered disk reads: 6 MB in  4.62 seconds =   1.30 MB/sec
```

“cached reads（高速缓存读）”表示 IDE 总线上的数据传输速率（和物理硬盘介质的吞吐量无关），而“buffered reads（缓冲读）”表示物理盘片的读开销。不出所料，物理硬盘要比总线慢得多。即便如此，1.3MB/s 的速度也太慢了。

既然 DMA 是现代硬件上的一种重要特性，它应该马上受到关注。比较新的硬盘支持几种 DMA 模式。表 7.2 总结了每种常见的 DMA 技术在各种 DMA 模式下的速度。

表 7.2　　各种 DMA 模式的最大传输速率（单位：MB/s）

模式	0	1	2	3	4	5	6
PIO	3.3	5.2	8.3	11.1	16.7		
SDMA	2.4	4.2	8.3				
MDMA	4.2	13.3	16.7				
UDMA	16.7	25.0	33.3	44.4	66.7	100.0	133.0

通过使用这张表，我们很容易就能确定传给 **hdparm** 的合适参数。我们通过输入下面的 **hdparm** 命令按 UDMA2 的模式启动 DMA：

```
$ sudo /sbin/hdparm -d1 -Xudma2 /dev/hdb

/dev/hdb:
```

www.TopSage.com

TopSage.com

```
    setting using_dma to 1 (on)
    setting xfermode to 66 (UltraDMA mode2)
    using_dma = 1 (on)
```

我们启用对 32 位 I/O 的支持：

```
$ sudo /sbin/hdparm -c1 /dev/hdb

/dev/hdb:
    setting 32-bit IO_support flag to 1
    IO_support = 1 (32-bit)
```

最后，消除中断屏蔽的值是二进制。我们用下面的命令打开它：

```
$ sudo /sbin/hdparm -u1 /dev/hdb

/dev/hdb:
    setting unmaskirq to 1 (on)
    unmaskirq = 1 (on)
```

我们可以重新运行 **hdparm** 的性能评测判断上述修改的作用：

```
$ sudo /sbin/hdparm -Tt /dev/hdb

/dev/hdb:
    Timing cached reads:        256 MB in 2.00 seconds = 127.83 MB/sec
    Timing buffered disk reads: 40 MB in 3.01 seconds = 13.30 MB/sec
```

只调整了几项 **hdparm** 的配置就取得了两方面性能的显著提升。

在硬盘上采用不同的参数值多次实验，找到最优的设置。在每种设置上运行几次性能评测，获得最准确的数据。

7.6 fsck: 检查和修复文件系统

现代的文件系统可靠性惊人，而且在发生出乎意料的系统崩溃以及古怪的硬件错误时，它会完成卓有成效的补救工作。不过，文件系统还是有多种可能被破坏或者变得不一致的。

每次发生内核失控或者电源失效的时候，都可能在崩溃之前给处于活动状态的文件系统引入少量的不一致性。因为内核对数据块和汇总信息进行缓冲，所以该文件系统最近的映像在硬盘和内存之间进行了分摊。在系统崩溃的过程中，映像的内存部分丢失了。内存中被缓冲的块被那些最近才保存到硬盘上的版本实际“覆盖”了。

解决这个问题有两个途径。小损坏通常可以使用 **fsck** 命令（“filesystem consistency check”，意思是文件系统一致性检查，要么大声读出它的拼法，要么发音为“fs check”或者“fisk”）来修复。从系统结构上来说，这并不是解决此问题的好方法，但对于所有常见的不一致性，它都能处理得非常好。

诸如 ReiserFS 和 ext3fs 这样的日志文件系统将元数据写出到一个连续的日志文件中，在命令返回之前日志文件才被冲洗（flush）到硬盘上。元数据最终会从日志中移到它在文件系统中的固定位置中去。如果系统崩溃了，那么日志就可以回滚到最近一次保持一致性的地方；不要求对文件系统进行完

全的交叉检查。在引导系统的时候，还要运行 **fsck**，以确保文件系统处于一致状态，但是它的运行速度比检查传统的 ext2 文件系统要快得多。在具有大型文件系统的系统中，这个功能可以帮您节省很多启动时间。

如果无法得到某种形式的日志，那么就必须等待 **fsck** 发挥它的威力。最常见的 5 种损坏类型是：

- 未被引用的 inode；
- 难以置信的超大链接数；
- 没有记录在磁盘块映射表中的未用数据块；
- 列出的空闲数据块还在某个文件中使用；
- 超级块中不正确的汇总信息。

fsck 能够安全和自动地修复这 5 类问题。如果 **fsck** 对文件系统进行了改正，那么您就应该反复运行它直到该文件系统能够完全干净地启动为止。

通常情况下，硬盘在启动时使用 **fsck -p** 来进行检查，它将检查**/etc/fstab** 中列出的所有本地文件系统，改正上面所列的 5 类错误。Linux 能知道哪些文件系统没有被干净地卸载，并且只检查那些“脏的”文件系统。如果启用了某些形式的日志功能，那么 **fsck** 只会告诉您该文件系统已经作了日志，并且将日志回滚到上次的一致状态中。

fsck -p 还能够在一个特定的文件系统上运行。比如说：

```
# fsck -p /dev/sda5
```

当 **fsck -p** 读取 **fstab** 文件来确定该检查哪些文件系统的时候，它遵守每个记录最后一个字段所指定的顺序，对文件系统按照数字的升序进行检查。如果两个文件系统在不同的硬盘上，那么可以给予它们相同的次序号。这种配置将使 **fsck** 同时检查它们，可以减少花费在等候硬盘 I/O 上的时间。系统总是首先检查根分区。

可以设置 Linux，在它已经安装某个文件系统若干次后（即使每次都“干净地”卸载这个文件系统）强制要求检查一遍这个文件系统。这个防范措施相当保险，在大多数情况下，默认值（通常为 20 次安装）足以满足要求。不过，在频繁安装文件系统的系统（比如桌面工作站）上，安装 20 次文件系统就执行一遍 **fsck** 可能会变得相当烦人。要把执行文件系统检查的间隔增加到 50 次安装，可以使用 **tune2fs** 命令，参见 tune2fs 的手册页了解几个作用稍小的 **tune2fs** 选项：

```
$ sudo /sbin/tune2fs -c 50 /dev/sda3
tune2fs 1.35(28-Feb-2004)
Setting maximal mount count to 50
```

不属于上面 5 种类型的错误可能很严重。它们会导致 **fsck -p** 命令先要求帮助，然后就退出。在这种情况下，不要带**-p** 选项运行 **fsck**。在手工模式下运行时，**fsck** 要求您去逐一确认它要进行的每项修复操作。下面的清单展示了一些 **fsck** 认为相当危险以致于需要人为加以干涉的错误。

- 不只一个文件声称拥有的磁盘块；
- 声称拥有的磁盘块在文件系统的范围之外；
- 过小的链接数；
- 不属于文件也不属于空闲块的磁盘块；
- 引用未分配 inode 的目录；
- 各种格式错误。

遗憾的是，如果没有对文件系统实现方面的广博知识，则很难通过手工修补硬盘。绝不要尝试通过设备文件来直接写文件系统。

实际上，这种状态意味着，除了去接受 **fsck** 所提出的解决方案，别无选择。通过仔细记录 **fsck**

www.TopSage.com

产生的消息可以将问题减到最少，因为这些消息有时候会提供有关引发问题的文件的线索。如果 **fsck** 请求获得权限去删除一个文件，那么就应该在允许 **fsck** 进行此项操作之前先将文件复制到一个不同的文件系统中。应该清楚，任何时候去试图访问一个已被损坏的文件系统，都会冒着使系统混乱的危险。

如果一个受损的文件系统（也就是 **fsck** 无法自动修复的文件系统）包含了非常有价值的数据，那么在稳固地备份它之前不要对其进行任何试验。您可以试着转储（**dump**）硬盘，但是因为 **dump** 希望读取的是一个没有损坏的文件系统而不是受损的文件系统，所以得到的映像可能会丢失数据（或者 **dump** 可能会崩溃）。最保险的策略就是 **dd** 整个硬盘到一个备份文件或者备份硬盘中去。参考第 9 章了解有关备份方法的知识。

如果 **fsck** 找到一个无法确定其父目录的文件，那么它将把该文件放置到文件系统顶级的 **lost+found** 目录中。因为一个文件的名字只会记录在该文件的父目录中，所以孤立文件没有名字，放在 **lost+found** 中的文件将用它们的 inode 号来命名。

7.7 向 Linux 增加硬盘：一步步的指导

在这一节里，我们会走一遍配置一个新硬盘的过程。我们给硬盘分几个分区，其中包括一个交换分区。我们在剩下的分区上创建 ext3fs 文件系统。

在安装好新硬盘之后，最好在启动内核之前先证实系统能够找到这个新设备。如果新设备是 IDE 硬盘，要检查确保在 BIOS 的设置显示中识别出了这个硬盘，通常在系统启动之前，键入一连串特殊键就能访问到 BIOS 的设置。参考随您的计算机或者主板带的手册，了解有关如何在 BIOS 中设置 IDE 设备的专门信息。在大多数情况下，不必进行特殊配置。

许多 SCSI 卡也有 BIOS 设置显示，您可以在系统启动之前把它调出来。如果有，您可以扫描 SCSI 总线，确认是否有新设备出现。如果这个过程中止或者产生了一条警告消息，那么可能是选了一个已经投入使用的 SCSI ID，或者是没有把终结器安装在正确的地方。

还可以使用 SCSI BIOS 对硬盘做低级格式化。在有的硬盘上，这项操作会花很长一段时间，而且不能被中断，所以做之前应该先做好计划。

如果您的 SCSI 卡没有提供它自己的用户界面，那么可以试着引导系统，并且注意内核显示出来的消息。如果没有看到任何来自 SCSI 驱动程序的消息，那么可能需要安装驱动程序，然后内核才能认出硬盘来。参考 28.6 节了解有关安装设备驱动程序的更多信息。

在我们的系统上，我们看到我们的 BusLogic SCSI 卡显示出了下面的消息：

```
scsi0 : BusLogic BT-948
scsi : 1 host.
  Vendor: SEAGATE   Model: ST446452W        Rev: 0001
  Type:   Direct-Access                     ANSI SCSI revision: 02
Detected scsi disk sda at scsi0, channel 0, id 3, lun 0
scsi0: Target 3: Queue Depth 28, Asynchronous
SCSI device sda: hdwr sector=512 bytes. Sectors=91923356 [44884 MB] [44.9 GB]
sda: unknown partition table
```

不必在意有关分区表的警告信息，因为这是硬盘第一次投入使用。一旦系统引导完毕，就可以开始对硬盘进行分区。

在对硬盘分区之前，必须首先检查是否已经有了该硬盘的设备文件（它们应该有设备文件）。在 Linux 上，SCSI 硬盘设备文件的名字采用**/dev/sd**XN 的形式，这里的 X 是一个小写字母，表示驱动器

www.TopSage.com

（字母 **a** 代表最小编号的 SCSI 硬盘，**b** 是编号第二小的 SCSI 硬盘，依此类推[7]），N 是分区号，从 1 开始进行编号。要表示整个硬盘，只要省略掉分区号就可以了。在 Linux 上，没有字符型（raw）的硬盘设备。

在本例中，我们的硬盘是 SCSI 链上的第一个硬盘。因此它的第一个分区就是**/dev/sda1**，而整个硬盘用**/dev/sda** 表示。如果没有这些设备文件，那么可以用 **MAKEDEV** 或者 **mknod** 创建它们。

现在硬盘已经准备就绪，可以进行分区了。和大多数的 PC 操作系统一样，Linux 下用来分区的工具叫做 **fdisk**。虽然 **fdisk** 的所有版本能做的事情都差不多（它们实现了微软的标准分区系统），但是它们中间还是有不少变化的。您最好读一下您所用的系统上的手册页，确保它同我们在这里介绍的 **fdisk** 相吻合。

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

The number of cylinders for this disk is set to 5721.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

既然我们把这块硬盘只用在我们的 Linux 系统上，所以我们会忽略帮助性的警告信息。以前曾经比较重要的一点是，让第一个分区比较小，即使用了比较老的 BIOS，也能保证工作，还能和系统上可能安装的其他操作系统和平共处。

fdisk 是一个交互式程序，按下 **m** 键会显示出它的所有命令。我们在这里会用到的有：

- **n** 或者 **new** 创建一个新分区；
- **t** 或者 **type** 改变分区的类型；
- **p** 或者 **print** 打印分配表；
- **w** 或者 **write** 把分区表写入硬盘。

因为我们的硬盘还没有分区，所以我们先创建一个新分区。如果硬盘原来有老分区，那么在创建新分区之前，必须先用 **fdisk** 的 **delete** 命令删除它们。用 **fdisk** 程序把分区表写入硬盘之前，它不会改变任何东西。

分区表可以容纳 4 个“主（primary）”分区，它们可以用来保存数据。另一种方法是创建一个“扩展（extended）”分区，它是一个主分区，指向了又一个分区表，能再有 4 个“逻辑（logical）”分区。虽然使用扩展分区能够克服正常情况下只有 4 个分区的限制，但如果仅仅需要不多的几个分区的话，只采用主分区仍是最简单的做法。我们在本例中就采用这个方法。

```
Command (m for help): new
Command action
   e   extended
   p   primary partition (1-4): p
Partition number (1-4): 1
First cylinder (1-5721, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-5721, default 5721): +2G
```

[7] 注意，这个字母是指 SCSI 设备目标号的顺序，不是目标号本身。如果添加或者去除一块硬盘，那么所有驱动器的字母都会改变！

www.TopSage.com

```
Command (m for help): print
Disk /dev/sda: 255 heads, 63 sectors, 5721 cylinders
Units = cylinders of 16065 * 512 bytes
   Device  Boot  Start   End     Blocks   Id  System
/dev/sda1            1   255   2048256   83  Linux
```

我们以相似的方式创建一个交换分区，唯一的不同在于我们把分区类型从 Linux 改为 Linux swap。虽然内核并不关注分区类型，但是有些程序或者脚本会试着用类型来判断每个分区是什么分区。目前，Linux swap 分区不能超过 2GB，所以我们设定了这个最大值。这可能超出了绝大多数应用的需要，但是既然我们有多余的硬盘空间，那么就可以表现得慷慨一些。如果 **mkswap** 不能使用我们分配给它的全部空间，那么它会向我们发出警告。

```
Command (m for help): new
  e   extended
  p   primary partition (1-4): p
Partition number (1-4): 2
First cylinder (256-5721, default 256): 256
Last cylinder or +size or +sizeM or +sizeK (256-1275, default 1275): 511
Command (m for help): type
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
```

第 3 个分区包含了硬盘剩下的全部空间，我们以类似的方式设定它。在写入分区表之前，我们最后再审核一遍它。

```
Command (m for help): print
Disk /dev/sda: 255 heads, 63 sectors, 5721 cylinders
Units = cylinders of 16065 * 512 bytes
   Device  Boot  Start   End     Blocks   Id  System
/dev/sda1            1   255   2048256   83  Linux
/dev/sda2          256   511   2056320   82  Linux swap
/dev/sda3          512  5721  41849325   83  Linux
```

如果分区不是在柱面边界处结束的，那么磁盘块数（blocks）旁边就会出现一个加号。您可以删除这个分区，（按照上面的显示）键入柱面数重新创建它，也可以保持原状，浪费一点儿硬盘空间。我们对新的分区表很满意，所以就把它写入了硬盘。

```
Command (m for help): write
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

现在我们已经为创建文件系统做好了准备。要创建一个 ext3fs 文件系统的话，只要运行 **mke2fs -j** 命令，并带上设备名作为参数。

```
# mke2fs -j /dev/sda1
mke2fs 1.36 (05-Feb-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
245280 inodes, 489974 blocks
...
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912
```

www.TopSage.com

大家网 TopSage.com

```
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
done
```

创建文件系统的时候，还可以使用**-J** 选项明确指定在新文件系统上日志文件的大小（**-J size=*x***），或者保存该日志文件的一个外部设备的标识（**-J device=*y***）。安装的时候一般把日志文件（大小必须介于 1 024 到 102 400 个文件系统块之间）放在文件系统里。

创建更大文件系统的步骤相同，但花费的时间要长得多。如果知道不需要 **mke2fs** 默认分配的那么多 inode，那么可以减少每簇的 inode 数，以此提高 **mke2fs** 的速度，并且留出更多的空间保存实际数据。同样，您或许希望增加文件系统的 inode 数，让它能保存大量的小文件。inode 太多要比太少好得多，因为用完了 inode 的话，就不能再创建更多的文件了。文件系统创建完毕之后，就不能增加更多的 inode 了。如果碰到了这样的情形，那么就需要把文件系统上的数据转储到磁带，或者是另一个分区的文件中，再以更大的 inode 数（**-i**）作为参数执行 **mke2fs**，然后用 **restore** 恢复这个分区上的数据。好好花几天时间干吧！

我们在我们的文件系统上运行 **fsck** 命令，以确保它们已经正确地创建好了。**-f** 标志迫使 **fsck** 一定检查新的文件系统，而不会假定它们已经清理干净了。

```
# fsck -f /dev/sda1
fsck 1.36 (05-Feb-2005)
e2fsck 1.36 (05-Feb-2005)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda1: 11/245280 files (9.1% non-contiguous), 16629/489974 blocks
```

新文件系统的安装点一创建好就可以安装该文件系统：

```
# mkdir /bkroot
# mount /dev/sda1 /bkroot
# df /bkroot
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/sda1              1981000        13   1878575   0% /bkroot
```

为了让系统在引导的时候安装新的文件系统，我们在文件**/etc/fstab** 中为每一个新文件系统加了一行。每行应该列出设备名、安装点、文件系统类型、安装选项、备份频度以及 **fsck** 检查的次序：

```
/dev/sda1       /bkroot         ext3    defaults        0 1
/dev/sda3       /new            ext3    defaults        0 2
```

为了让硬盘设备能够引导，必须在它上面写入引导加载程序。根据您安装的系统情况的不同，实际安装引导加载程序的命令要么是 **lilo**，要么是 **grub**。参考 2.3 节了解有关配置和安装一个引导加载程序的更多信息。

最后一步是创建交换空间并把它加入系统。交换分区必须用 **mkswap** 进行初始化，该命令以分区的设备名作为参数，不必再指定交换分区的大小了。创建好了交换分区之后，我们用 **swapon** 命令启用这个分区。**swapon** 还会验证交换区的加载是否正确。

www.TopSage.com

```
# mkswap /dev/sda2
Setting up swapspace version 1, size = 2105667584 bytes
# swapon /dev/sda2
# swapon -s
Filename    Type        Size      Used   Priority
/dev/hda5   partition   133020    688    -1
/dev/sda2   partition   2056316   0      -2
```

和普通的文件系统一样，如果您想让系统在下次启动的时候记住新的交换分区，那么必须把它也加入到**/etc/fstab** 文件里。下面的配置行适用于我们的硬盘：

```
/dev/sda2          swap          swap          defaults      0  0
```

最后重启系统测试对**/etc/fstab** 文件所做的修改是否正确，还要确保新的文件系统和交换空间都正确地挂到系统上了。

7.8 高级硬盘管理：RAID 和 LVM

迄今为止，我们已经讨论过添加新硬盘、新文件系统和新交换区的步骤，它们和大多数 UNIX 系统上使用的步骤都类似。不过，Linux 还有一些特色功能，对于许多操作系统来说，这些功能还只是一个梦想。软 RAID 和逻辑卷管理这两种独特的技术给 Linux 的硬盘管理多增加了一层灵活性和可靠性。

硬盘会频繁出现故障，即便当前有备份，在服务器上出现硬盘故障造成的后果也是灾难性的。RAID（独立磁盘的冗余阵列，Redundant Arrary of Independent Disks）是一个使用多个硬盘，在几块硬盘上分散存储或者重复存储数据的系统。RAID 不仅有助于避免数据丢失，还能把因为硬件故障造成的停机时间减少到最小（通常降低到 0），同时还有可能提高性能。RAID 系统可以由硬件实现，但是 Linux 系统用软件也实现了 RAID 所有必要组件。

第二个同样有用的工具叫做 LVM（逻辑卷管理，logical volume management），它帮助系统管理员高效地在多个分区之间分配可用的硬盘空间。设想一下这样的情形：管理员没有准确地掌握一个分区确实需要多大，在创建分区半年之后发现这个分区太大了，可是相邻的分区却没有足够的空间。听上去是不是有点熟悉？LVM 把空间动态地从富余的分区重新分配给不足的分区。

虽然这两种工具单独使用起来功能强大，但将它们组合使用尤其有效。本节下面就介绍两种系统的概念，并且举例说明具体的配置。

7.8.1 Linux 的软 RAID

最近我们在一台重要的生产服务器上遭遇了一次硬盘控制器故障。虽然数据是分布在几块物理硬盘上进行复制的，但是发生故障的硬件 RAID 控制器毁掉了所有硬盘上的数据。随后便经历了从磁带恢复系统的漫长而难耐的过程，大约两个月之后服务器才彻底恢复。重建的服务器现在依靠内核的软件来管理 RAID 环境，借此消除了再次发生 RAID 控制器故障的可能性。

RAID 能够做两件基本的事情。首先，RAID 通过把数据分散到多个硬盘上，因而能让多个硬盘同时工作提供或者读取一条数据流，所以可以提高性能。其次，RAID 在多个硬盘上复制或者“镜像”数据，降低了由于一块硬盘发生故障带来的风险。Linux 的 RAID 和传统的 RAID 有些细微的不同之处，但是它在逻辑上仍然要分成几级。

www.TopSage.com

- 线性模式没有提供数据冗余性或者提高性能。这个模式仅仅把多个硬盘的块地址连到一起创建一个（更大的）虚拟硬盘。
- RAID 0 只能提高性能。这个模式使用两个或者多个容量相同的硬盘来缩短写数据和访问时间。
- RAID 1 是第一个能提供冗余性的级别。它同时在两个或者多个硬盘上复制数据。这个模式镜像数据但却降低了性能，因为必须不止一次写数据。
- RAID 4 能和 RAID 5 竞争，但一贯不如 RAID 5。这个模式把数据分片存储，但是需要用一个硬盘保存校验信息，因此会花时间等待写校验硬盘。除非有非常正当的理由使用 RAID 4，否则不要管它，先选 RAID 5。
- RAID 5 是 RAID 的“世外桃源”。通过把数据和校验信息都分片存储，这个模式既创建了一种冗余结构，又同时降低了读写时间。RAID 5 需要至少 3 块硬盘。

Linux 内核从 2.0 版开始内建软 RAID 功能，但是早期的内核版本有 bug，而且功能不完善。我们建议避开 2.4 版内核之前的实现。

7.8.2 管理逻辑卷

LVM（Logical Volume Manager，逻辑卷管理器）是一种可选的子系统，它提供了硬盘分区的一种超强版本。它能让您将多个单独的硬盘组成“卷组（volume group）”。卷组的总容量接着可以分配给逻辑卷（logical volume），后者按照普通的块设备来访问。逻辑卷的管理机制能让您：

- 更高效地使用和分配硬盘存储空间；
- 在不同的物理设备之间移动逻辑卷；
- 随时扩大和缩小逻辑卷的大小；
- 保持整个文件系统的“快照”；
- 不必中断设备就能替换联机设备。

一个逻辑卷的各个组成部分能够以不同的方式组合在一起。连接方式（Concatenation）把每个设备的物理块放到一起，然后一个接一个地连起来。条带方式（Striping）将各个部分交错分布，相邻的虚拟块实际上散布在多个物理硬盘上。条带方式通过降低单硬盘的瓶颈，往往可以提供更高的带宽、更低的延迟。

7.8.3 配置 LVM 和 RAID 的例子

我们前面举的例子说明了一个基本硬盘的配置。在这一节，我们讲一遍包括 RAID 和 LVM 的配置步骤。对于生产环境的服务器来说，这种配置特别有用。

我们的目的是在 3 个空硬盘上创建一个 RAID 5 盘阵。在这个 RAID 盘阵上划定两个 LVM 分区，web1 和 web2。比起传统的系统，这样的结构有以下几个优势。

- RAID 5 有冗余性，如果硬盘发生故障，数据仍然能够访问。RAID 5 和 RAID 4 不一样，硬盘发生故障没关系。
- 有了 LVM，分区的大小可以改变。当一个热衷于网站的 Web 管理员用完了 web2 之后，我们可以方便地从 web1 多借一些空间。
- 两个分区最终还是会要更多的硬盘空间。这样的设计能向 RAID 5 盘阵再添加硬盘。一旦完成这样的设置，现有的 LVM 组就可以扩展包括进来更多的空间，完全不需要再创建任何分区。

在给出初始的配置之后，我们再介绍如何处理一块硬盘发生故障的情况，以及如何改变 LVM 分区的大小。

在我们举例的系统上，有四块大小相等的 SCSI 硬盘：

www.TopSage.com

```
# fdisk -l

Disk /dev/sda: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device   Boot   Start         End      Blocks      Id      System
/dev/sda1      *       1          13      104391      83      Linux
/dev/sda2     14     144     1052257+        82      Linux   swap
/dev/sda3    145    2213    16619242+        8e      Linux   LVM1G
Disk /dev/sdb: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device   Boot   Start         End      Blocks      Id      System

Disk /dev/sdc: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device   Boot   Start         End      Blocks      Id      System

Disk /dev/sdd: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device   Boot   Start         End      Blocks      Id      System
```

第一块 SCSI 硬盘**/dev/sda** 上有我们的系统分区。其他 3 块硬盘（**sdb**、**sdc** 和 **sdd**）都没有分区表。

首先，我们在每块 SCSI 硬盘上创建分区。因为硬盘都是一样的，所以在每块硬盘上执行的命令都相同。

```
# fdisk /dev/sdb
...
Command (m for help): new
Command action
  e   extended
  p   primary partition (1-4): p
Partition number (1-4): 1
First cylinder (1-2213, default 1): <Enter>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-2213, default 2213): <Enter>
Using default value 2213

Command (m for help): type
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): write
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

再写好另外两块硬盘的分区卷标之后，就要开始建立 RAID 盘阵。大多数现代的发行版本都使用 **mdadm** 命令来管理 RAID。RHEL 以前的版本使用 **raidtools** 软件，但是因为 **mdadm** 命令比 **raidtools** 功能更强，使用起来更容易，所以我们在这里就用 **mdadm** 来讲解。

下面的命令利用 3 个 SCSI 分区建立一个 RAID 5 盘阵：

www.TopSage.com

```
# mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdc1
    /dev/sdd1
mdadm: array /dev/md0 started.
```

在建盘阵的时候，**/proc/mdstat** 文件显示出的进度信息如下：

```
# cat /proc/mdstat
Personalities : [raid5]
md0 : active raid5 sdb1[3] sdc1[1] sdd1[2]
35566336 blocks level 5, 64k chunk, algorithm 2 [3/2] [_UU]
    [====>.............]  recovery = 22.4% (3999616/17783168) finish=5.1min
    speed=44800K/sec
unused devices: <none>
```

这个文件总是反映着内核的 RAID 系统当前的状态。在添加新硬盘或者替换故障硬盘的时候，观察这个文件非常有用（**watch cat /proc/mdstat** 命令就是一个方便的方法）。

盘阵的组建工作完成之后，我们会在**/var/log/messages** 文件里看到下面的通知消息：

```
RAID5 conf printout:
--- rd:3 wd:3 fd:0
disk 0, o:1, dev:sdb1
disk 1, o:1, dev:sdc1
disk 2, o:1, dev:sdd1
```

刚开始创建盘阵的命令也会“启动”盘阵（让它能够使用），但是以后重启系统的时候我们需要单独有一步启动盘阵，通常这一步放在启动脚本里。RHEL、Fedora 和 SUSE 都带有 RAID 启动脚本的例子。

虽然给 **mdadm** 命令一个配置文件它也会用（一般是**/etc/mdadm.conf**），但是就技术而言，它并不要求有一个配置文件。我们强烈建议使用一个配置文件。它按一种标准方式说明了 RAID 的配置，因而给了系统管理员一个在出现问题的时候可以查看的明确的地方。另一种使用配置文件的方法是在每次启动盘阵的时候在命令行指定配置文件。

mdadm --detail --scan 命令把当前的 RAID 设置存储到一个配置文件里。遗憾的是，该命令打印出的配置不是非常完整。下面的命令能为我们举例的设置提供一个完整的配置文件：

```
# echo DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1 > /etc/mdadm.conf
# mdadm --detail --scan >> /etc/mdadm.conf
# cat /etc/mdadm.conf
DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1
ARRAY /dev/md0 level=raid5 num-devices=3 UUID=21158de1:faaa0dfb:
    841d3b41:76e93a16
  devices=/dev/sdb1,/dev/sdc1,/dev/sdd1
```

mdadm 命令现在在启动或者停止的时候会读这个文件，这样一来可以方便地管理盘阵。为了用新建的**/etc/mdadm.conf** 文件在系统启动的时候启用盘阵，我们执行：

```
# mdadm -As /dev/md0
```

要手工停止盘阵，我们使用下面的命令：

```
# mdadm -S /dev/md0
```

现在我们已经把 3 块硬盘组建成了一个逻辑上的 RAID 硬盘。现在就可以在我们创建的可扩大（也可以缩小）的文件系统上划定逻辑卷组了。配置 LVM 有几个清楚的步骤：

- 创建（实际上是划定）和初始化物理卷；

TopSage.com

- 把物理卷加入一个卷组；
- 在卷组上创建逻辑卷。

LVM2 工具软件能解决所有这些任务，而且能方便以后管理卷。**man lvm** 很好地介绍了这个系统及其工具。

按照 LVM 的术语，物理卷是聚集到一起形成存储池（逻辑卷）的东西。不过，物理卷是个有点误导性的术语，因为物理卷不需要直接对应物理设备。它们可以是硬盘，但它们也可以是硬盘分区（本例就是），或者是有自己的底层结构的高级 RAID 对象。

LVM 命令以字母开头，字母反映了它们在哪个抽象层面上执行操作：**pv** 命令操作物理卷，**vg** 命令操作卷组，而 **lv** 命令操作逻辑卷。

LVM 比较老的版本要求第一步先执行 **vgscan** 命令，但是现在不必要了。代之以直接用 **pvcreate** 命令初始化每个物理设备。对于本例来说，我们使用我们刚创建的 RAID 5 设备**/dev/md0**。

```
# pvcreate /dev/md0
Physical volume "/dev/md0" successfully created
```

这个操作会破坏设备或者分区上的所有数据，所以我们要特别特别小心！虽然我们在这个例子里只用了一个物理设备，但是 LVM 允许我们向一个卷组增加多个不同类型的设备。

我们的物理设备现在准备好了被加入到一个卷组：

```
# vgcreate LVM1 /dev/md0
Volume group "LVM1" successfully created
```

要回头检查一下我们的手艺，使用 **vgdisplay** 命令：

```
 # vgdisplay LVM1
 --- Volume group ---
 VG Name               LVM1
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               33.92 GB
PE Size               4.00 MB
Total PE              8683
Alloc PE / Size       0 / 0
Free                  PE / Size8683 / 33.92 GB
VG UUID               nhkzzN-KHmY-BfV5-6F6Y-3LF8-dpd5-JM5lMp
```

最后几步要在卷组 LVM1 里创建逻辑卷，再在卷上创建分区。我们让两个逻辑卷每个都 10GB 大：

```
# lvcreate -L 10G -n web1 LVM1
Logical volume "web1" created

# lvcreate -L 10G -n web2 LVM1
Logical volume "web2" created
```

www.TopSage.com

既然我们已经在卷组 LVM1 里创建了两个逻辑卷 web1 和 web2，我们就能创建并安装我们的文件系统。

```
# mke2fs -j /dev/LVM1/web1
...
# mke2fs -j /dev/LVM1/web2
...
# mkdir /web1 /web2
# mount /dev/LVM1/web1 /web1
# mount /dev/LVM1/web2 /web2
```

文件系统最后准备就绪，可以投入使用。我们把新文件系统加到**/etc/fstab** 文件，重启系统保证所有配置都能运行无误。

7.8.4 处理故障硬盘

现在我们仔细组建的系统看上去挺不错的，但是系统运行在多个层面上，所以匆忙间事情就可能出现问题。当硬盘出现故障或者分区受损（或者只是分区填满了）的时候，需要知道怎样很快和很容易地修好。管理员可以使用和前面进行初始配置一样的工具来维护系统，从故障中恢复。

考虑一个硬盘发生故障的情况。因为 RAID 5 提供了某种程度的数据冗余性，我们在上一节组建的 RAID 5 盘阵在一块硬盘发生故障的时候仍然能继续工作，用户不必知道任何问题。管理员需要仔细检查系统日志，及早发现问题（或者让一个程序替您做这件事情，参见 10.5 节）。

mdadm 命令提供了一个方便的选项能模拟一个硬盘发生故障：

```
# mdadm /dev/md0 -f /dev/sdc1
mdadm: set /dev/sdc1 faulty in /dev/md0

# tail /var/log/messages
May 30 16:14:55 harp kernel: raid5: Disk failure on sdc, disabling device.
    Operation continuing on 2 devices
kernel: RAID5 conf printout:
kernel:  --- rd:3 wd:2 fd:1
kernel:  disk 0, o:1, dev:sdb1
kernel:  disk 1, o:0, dev:sdc1
kernel:  disk 2, o:1, dev:sdd1
kernel: RAID5 conf printout:
kernel:  --- rd:3 wd:2 fd:1
kernel:  disk 0, o:1, dev:sdb1
kernel:  disk 2, o:1, dev:sdd1
```

如上所示，一旦发生（模拟的）故障，系统日志**/var/log/messages** 里马上就有相关信息。从 RAID 的状态文件**/proc/mdstat** 也能得到类似信息。此刻系统管理员应该采取如下措施：

- 从 RAID 盘阵中删除硬盘；
- 安排停运时间，关闭计算机（如果必要的话）；
- 更换物理硬盘；
- 向盘阵加入新硬盘。

要从 RAID 的配置里删除设备，使用 **mdadm**：

```
# mdadm /dev/md0 -r /dev/sdc1
mdadm: hot removed /dev/sdc1
```

硬盘一旦在逻辑上被删除了，就可以更换硬盘。支持热交换的硬件让用户可以不必关闭系统或者

www.TopSage.com

重启就能更换硬盘。

如果直接用硬盘组建 RAID，那么应该只用相同的硬盘更换它们。用分区组建的 RAID 可以用相同大小的分区来替换，虽然为了带宽匹配的缘故，如果硬件上能采用相似的硬盘那就最好了。（如果 RAID 的配置是在分区上组建的，在向盘阵加入替换的硬盘之前，必须运行 **fdisk** 正确地划定分区。）

在我们的例子中，只是模拟的故障，所以我们不必更换任何硬件就能直接把硬盘加回盘阵：

```
# mdadm /dev/md0 -a /dev/sdc1
mdadm: hot added /dev/sdc1
```

Linux 重建盘阵，并且还会一如既往地在**/proc/mdstat** 文件里反映重建的进度。

7.8.5 重新分配存储空间

用户或者日志文件填满了分区是比硬盘故障更常见的情况。从保存个人 MP3 音乐的服务器到装满计算机系邮件的服务器，我们在这些服务器上经历了所有的故障情况。

假如我们所举例子里的**/web1** 增长超出了我们的预期，它需要更多空间。只要简短几步，就可以调整 LVM 分区的大小。到底用什么样的命令取决于所用的文件系统。下面这个例子里采用的步骤针对的是 ext3 文件系统。这些步骤包括：

- 研究当前的 LVM 配置；
- 用 **lvextend** 和 **ext2online** 命令调整分区的大小；
- 核实调整是否到位。

幸好我们在卷组里多留了一些空间，让**/web1** 能用这些空间扩展，所以我们用不着从别的卷搜刮空间。我们用 **vgdisplay** 看一下卷组里的可用空间，用 **df** 判断如何重新分配空间：

```
# vgdisplay LVM1
--- Volume group ---
VG Name               LVM1
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  3
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                2
Open LV               2
Max PV                0
Cur PV                1
Act PV                1
VG Size               33.92 GB
PE Size               4.00 MB
Total PE              8683
Alloc PE / Size       5120 / 20.00 GB
Free                  PE / Size3563 / 13.92 GB
VG UUID               nhkzzN-KHmY-BfV5-6F6Y-3LF8-dpd5-JM5lMp

# df -h /web1
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/LVM1-web1 9.9G  7.1G  2.3G  76% /web1
```

这些命令显示出卷组里还有 13.92GB 的空闲空间，而**/web1** 的使用率达到 76%。我们给**/web1** 加上 10GB。首先，我们用 **lvextend** 把空间加入逻辑卷，然后再用 **ext2online** 扩大文件系统结构，把增加的空

www.TopSage.com

间包括进来。

```
# lvextend -L+10G /dev/LVM1/web1
  Extending logical volume web1 to 20.00 GB
  Logical volume web1 successfully resized
# ext2online -d /dev/LVM1/web1
...
ext2_close
```

这就对了！再次考察 **df** 的输出反映出了变化：

```
# df -h /web1
Filesystem            Size  Used  Avail  Use%  Mounted on
/dev/mapper/LVM1-web1  20G  7.1G   12G   38%  /web1
```

ReiserFS 的用户必须在运行 **lvextend** 命令前先卸载分区。此外，用一个名为 **resize_reiserfs** 的工具代替 **ext2online** 改变文件系统的大小。

7.9 安装 USB 设备

软盘终于已经绝迹，太好了，总算走了。取代它的是友好、快速而且有意思的 USB 驱动器。这些设备有多种形式：个人 U 盘、数码相机、iPod 和大容量外置硬盘是我们能叫得出的几个名字。最近的 Linux 发行版本原配的内核就支持所有这些方便的小发明。

连接一个 USB 驱动器的时候，首先要确认 Linux 能够识别这个设备。下面的命令列出了内核已经发现的 USB 设备：

```
$ sudo /sbin/lsusb
Bus 001 Device 003: ID 0781:7108 SanDisk Corp.
Bus 001 Device 001: ID 0000:0000
```

在这个例子里，接了一个 SanDisk 驱动器。如果设备插上了，可没有设备列出，内核可能不支持 USB，需要重新编译内核。

接下来，搞清楚内核怎样标识这个驱动器，内核用什么设备文件表示这个驱动器。内核的消息通过 syslog 记录了下来。

```
$ sudo tail -n 20 /var/log/messages | grep kernel[8]
Jul 27 20:52:13 harp kernel: USB Mass Storage support registered.
Jul 27 21:02:57 harp kernel: usb 1-2: USB disconnect, address 2
Jul 27 21:14:09 harp kernel: ohci_hcd 0000:00:0f.2: wakeup
Jul 27 21:14:09 harp kernel: usb 1-2: new full speed USB device using addr 3
Jul 27 21:14:09 harp kernel: scsi3 : SCSI emulation for USB Storage devices
Jul 27 21:14:09 harp kernel: Vendor: SanDisk Model: Cruzer Titanium Rev: 2000
Jul 27 21:14:09 harp kernel: Type: Direct-Access ANSI SCSI revision: 02
Jul 27 21:14:09 harp kernel: SCSI device sde: 512-byte hdwr sectors (520 MB)
Jul 27 21:14:09 harp kernel: sde: Write Protect is off
Jul 27 21:14:09 harp kernel: sde: assuming drive cache: write through
Jul 27 21:14:10 harp kernel: sde: sde1
Jul 27 21:14:10 harp kernel: Attached scsi removable disk sde at scsi3, channel
    0, id 0, lun 0
```

[8] 在 Debian 和 Ubuntu 的系统上要看/var/log/kern.log。

www.TopSage.com

内核消息表明这是一个520MB的SanDisk Cruzer Titanium(如果要买的话,它是一种很棒的U盘)。内核把**/dev/sde**这个设备同该盘关联起来,这个U盘只带一个分区**sde1**。

Linux在能使用这个U盘之前先要安装它。创建一个安装点,再安装U盘:

```
$ sudo mkdir /mnt/usb
$ sudo mount /dev/sde1 /mnt/usb
```

U盘现在安装在**/mnt/usb**,可以用了。为了简化下次用这个U盘时的过程,可以在**/etc/fstab**中加上下面一行:

```
/dev/sde1 /mnt/usb auto users,noauto,uid=ben,gid=users  0   0
```

其中列出的选项会自动检测文件系统的类型,允许用户ben安装和写这个U盘。

7.10 习题

E7.1 列出SCSI的所有种类。每种SCSI该用什么样的连接器?不考虑连接器的差异,不同的SCSI版本之间有哪些兼容性的问题?

E7.2 硬盘格式化和硬盘分区之间有什么区别?分区和创建文件系统之间有什么区别?

E7.3 列出在以下环境中,在一个硬盘上创建一个文件系统所要用到的命令和参数。

a)硬盘用于保存home目录;

b)硬盘用于交换区;

c)硬盘用在一个很大的垃圾邮件站点存储邮件队列;

d)硬盘装有一个MySQL InnoDB数据库。

E7.4 LVM工具软件功能强大,但是如果没有搞得很清楚就可能发生混淆。实际联系一下在一个卷组内增加、删除硬盘,或者调整硬盘大小。说明如何成功地从一个卷组中删除一个设备,并将它加入另一个卷组。

★E7.5 使用书面印刷的或者Internet上的资源,确定性能最好的SCSI硬盘和IDE硬盘。评测这些硬盘所采用的评测指标能反映出一台繁忙的Linux服务器把它用作引导盘时的情况吗?您会为SCSI多花多少钱?您从多花的钱上得到了多高的性能提升(如果有的话)?

★E7.6 给您的系统增加一块硬盘。让新硬盘的一个分区备份根分区,安装一个内核并从该分区引导。记录下完成此项任务所需的所有步骤。您可能发现脚本很有帮助(需要root权限)。

★E7.7 什么是超级块,它有什么用途?在内核的头文件中查出ext2fs超级块的结构定义,讨论结构中每个域所代表的含义。

★E7.8 使用**mdadm**及其**-f**选项模拟RAID盘阵中一块硬盘发生故障。从这个盘阵中删除该硬盘,再把它加回去。每一步**/proc/mdstat**显示怎样的信息?

★★E7.9 ext3fs文件系统上的inode里保存有哪些信息域?列出代表文件**/etc/motd**的inode的内容。这个文件的文件名保存在什么地方(诸如**hexdump**和**ls -i**这样的命令可能会有帮助)?

★★E7.10 用诸如**hexdump**或者**od**这样的程序研究一个目录文件的内容。每个长度可变的记录代表目录中的一个文件。查看目录项的结构,用一个真实的目录文件为例,说明每个域的作用。然后,看一下任何文件系统上的**lost+found**目录。为什么**lost+found**

www.TopSage.com

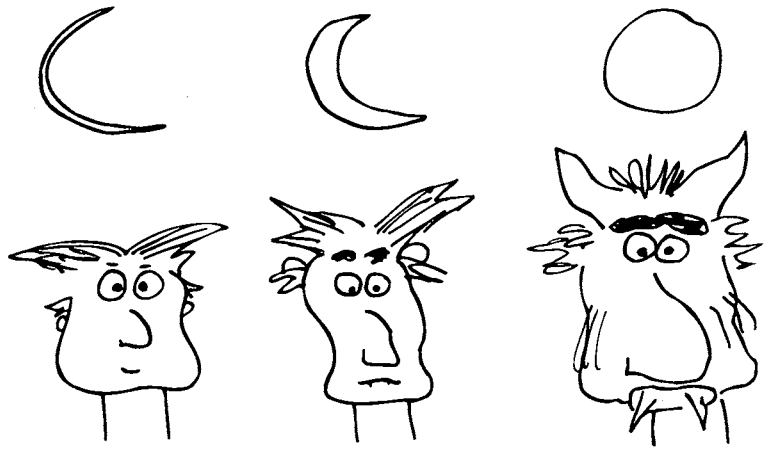
大家网 TopSage.com

里有那么多名字，但是这个目录看上去却是空的？

★★★★★E7.11 编写一个程序，遍历文件系统，打印出**/etc/motd** 和**/etc/termcap** 文件的内容。但是不要直接打开这两个文件，而是打开根分区的原始设备文件，用系统调用 **seek** 和 **read** 解读文件系统，找出确切的数据块。**/etc/motd** 通常不长，而且可能只包含直接的磁盘块（direct block），**/etc/termcap** 可能要您解读间接的磁盘块（indirect block）。提示：在阅读系统头文件的时候，要确信您已经找到了文件系统在磁盘上的索引结点结构，而不是在内存中的索引结点结构（需要 root 权限）。

www.TopSage.com

第 8 章　周期性进程



让系统保持在系统管理员掌握之中的关键，就是让尽可能多的任务自动完成。例如，**adduser** 程序添加新用户的速度比您更快，而且出错几率更小。几乎任何一个任务都能用 Perl、Python、shell 或者 **expect** 脚本编写。

让一个脚本或者命令在没有任何人为干预的情况下执行往往是非常有用处的。例如，您可能需要让一个脚本去验证（比如说，每隔半个小时验证一次）您网络中的路由器和桥接器都在正确工作，并且在发现问题的时候让它给您发电子邮件[1]。

8.1　cron:按时间表执行命令

在 Linux 系统下，周期执行的任务一般由 **cron** 这个守护进程来处理。**cron** 在系统引导时启动，并且只要系统在工作，它就会保持运行。**cron** 读取一个或者多个配置文件，这些配置文件中包含了命令行及其调用时间的清单。命令行是由 **sh** 来执行的，所以，能够手工从 shell 中完成的任何操作差不多都能通过 **cron** 来完成[2]。

在 20 世纪 70 年代，**cron** 最初出现在 UNIX 的大家族中。Linux 发行版本所带版本称为 ISC **cron**，或者 Vixie-**cron**，这是因其作者 Paul Vixie 而得名的。Vixie-**cron** 是重新编写的现代版本，它既提供了更多的功能，又减少了麻烦。

cron 的配置文件称为“crontab”，它是“cron table”的缩写。**cron** 在 3 个地方查找 crontab 文件：**/var/spool/cron**（SUSE 上是**/var/spool/cron/tabs**，而在 Debian 上是**/var/spool/cron/crontabs**）、**/etc/cron.d** 和**/etc/crontab**。

[1] 很多站点做得都比这更进了一步，一旦发现问题，它们就会向管理员的呼机发送一则文本消息。

[2] 事实上，Linux 下的**/bin/sh** 实际是到 **bash** shell 的一个链接，它是 UNIX 系统上传统的 Bourne shell 的增强（和重新实现）版本。用户也可以配置 **cron** 使用其他的 shell。

www.TopSage.com

每个用户的 crontab 文件都保存在**/var/spool/cron** 目录下。一般而言，一个用户（最多）有一个 crontab 文件：一个 crontab 文件给 root 用户，一个 crontab 文件给 jsmith，依此类推。crontab 文件是以它们所属用户的登录名来命名的，**cron** 使用这些文件名来决定在运行每个文件所包含的命令时应该使用哪一个 UID。**crontab** 命令把 crontab 文件传入和传出这个目录。

负责安排由系统管理员制定的系统维护以及其他任务的 crontab 文件，都保存在**/etc/crontab** 文件以及**/etc/cron.d** 目录下的其他文件里。这些文件的格式同每个用户自己的 crontab 文件格式略有不同，因为它们允许以任何用户身份来执行命令。**cron** 以完全一样的方式对待**/etc/crontab** 文件和**/etc/cron.d** 目录下的文件。一般而言，**/etc/crontab** 供系统管理员手工维护，而**/etc/cron.d** 目录则提供了一个地方，软件包把它们可能需要的任何 crontab 项安装在这里。

当 **cron** 启动的时候，它会读取它的所有配置文件，把它们保存在内存中，然后转入睡眠状态。每一分钟，**cron** 都会醒来一次，检查 crontab 文件的修改时间，重新载入任何已被改过的文件，然后在返回睡眠状态之前，执行为这一分钟安排的所有任务。



不知道什么原因，Red Hat 系统上的 **cron** 已经改名叫 **crond**。

cron 通过 syslog 为它的活动做日志，采用的功能为“cron”，大多数消息的提交级别为“info”。默认的 syslog 配置一般把 **cron** 的日志数据发送到它自己的文件。

8.2 crontab 文件的格式

一个系统上的所有 crontab 文件都使用一种类似的格式。在每一行的第一列中可以使用“#”号来引入注释。每个非注释行包含了 6 或 7 个字段，它代表了一条命令：

```
minute hour day month weekday [username] command
```

前 6 个字段用空白隔开，但在 *command* 字段中的空白则按照原意对待。*username* 字段只出现在**/etc/crontab** 文件和**/etc/cron.d** 目录下的文件里，它指出命令应该以哪个用户的身份运行。这个字段不会也没有必要出现在用户自己的 crontab 文件中（那些保存在**/var/spool/cron** 下的文件），因为文件名本身就暗含了 UID。

minute、*hour*、*day*、*month* 和 *weekday* 给出了关于命令应该何时运行的信息。它们的解释在表 8.1 中给出。

表 8.1 **crontab 的时间规范**

字　段	描　述	范　围
minute	小时中的分钟	0～59
hour	天中的小时	0～23
day	月中的天	1～31
month	年中的月	1～12
weekday	星期中的天	0～6（0=星期天）

每个和时间相关的字段可以包含：

- 星号，它可以匹配所有字符；
- 一个整数，它必须精确匹配；
- 用短划线隔开的两个整数，它们匹配的是值的范围；
- 一系列用逗号隔开的整数或者范围，它们匹配任何被列出的值。

www.TopSage.com

时间范围可以包含一个步长值。例如，时间序列 0、3、6、9、12、15、18 可以更简洁地写成 0-18/3。月份和日期的名字也可以使用易记的文字名，但是此时不能连同“表示范围的方式”一起使用。

时间格式

```
45   10   *    *   1-5
```

的意思是“上午 10:45，从周一到周五”。我们的提示是：不要把星号放在第一个字段中，除非您想让这个命令每分钟都运行。

注意 *weekday* 字段和 *day* 字段有潜在的二义性。每一天既是星期中的一天，也是月中的一天。如果同时指定了 *weekday* 字段和 *day* 字段，满足两个条件之一的天就被选中。例如，

```
0,30    *    13   *    5
```

的意思是“星期五每半个小时，以及每月 13 号每半个小时”，而不是“既是 13 号又是星期五的那天每半个小时”。

command 就是要执行的 **sh** 命令行。它可以是任何有效的 shell 命令，而且不应该加引号。**cron** 认为 *command* 一直是到这行的末尾，它可以包含空格或者制表符。

cron 使用百分号（%）来表示 *command* 字段中的换行。只有到第一个百分号之前的文本才会包含在实际命令中，其余行则作为该命令的标准输入。

这里有一些合法 crontab 命令的例子：

```
echo The  time  is  now 'date' > /dev/console
mail -s Reminder evi@anchor % Don't forget to write your chapters.
cd /etc; /bin/mail -s "Password file" evi  < passwd
```

这还有一些关于 contrab 项的完整的例子。

```
30  2  *  *  1    (cd  /users/joe/project;  make)
```

这一行将在每个星期一的凌晨 2:30 时被激活。它会在**/users/joe/project** 目录中运行 **make** 命令。像这样的一行会在其他用户不使用系统时，启动一个很长的编译过程。通常情况下，**cron** 命令产生的任何输出都会被寄给 crontab 的“属主”[3]。

```
20  1  *  *  *     find /tmp -atime +3 -exec rm -f { }  ';'
```

这条命令将在每天凌晨 1:20 运行。它将删除在**/tmp** 目录中 3 天内没有被人访问过的所有文件。

```
55   23   *    *   0-3,6  /staff/trent/bin/checkservers
```

这一行将在除了星期四和星期五之外的每天晚上 11:55 时运行 **checkserver**。

在 crontab 文件里也可以指定环境变量和它们的值。参考手册页 **crontab**（5）了解更多细节。

8.3 crontab 管理

crontab *filename* 把 *filename* 安装为 crontab 文件，它将替换 crontab 文件的任何以前版本。**crontab -e** 检出 crontab 的一个副本，调用编辑器（由环境变量 EDITOR 所指定）打开它，然后将其重新提交给 crontab 目录。**crontab -l** 将 crontab 中的内容列在标准输出上，**crontab -r** 将会删除它，不给您留下

[3] 也就是 crontab 文件名所指的用户。在大多数（但并非所有的）系统上，crontab 文件的实际属主一般是 root。

www.TopSage.com

crontab 文件的一点儿内容。

root 可以给 crontab 带一个 *username* 参数，这样就能够查看或编辑其他用户的 crontab 文件了。例如，**crontab -u jsmith-r** 将删除属于用户 jsmith 的 crontab 文件。

在不带命令行参数的情况下，大多数版本的 **crontab** 将试图从它自己的标准输入中读取 crontab 的内容。如果您意外地进入了这种模式，那么不要试图使用<Control-D>来退出该模式，这样做将会删除您的整个 crontab 内容。代之以使用<Control-C>退出。有些版本的 **cron** 经过修改，要求用户提供一个短划线（-）作为文件名参数，让 crontab 从标准输入读取 crontab 的内容。

用**/etc/cron.allow** 和**/etc/cron.deny** 这两个配置文件可以指定哪些用户可以提交 crontab 文件。如果存在 **cron.allow** 文件，那么它将包含可以提交 crontab 的所有用户的名单，名单的每行一个用户。任何没被列出的用户都不能调用 **crontab** 命令。如果没有 **cron.allow** 文件，那么将检查是否有 **cron.deny** 文件。它也是一个用户的名单，但是意思却反过来了：除了名单中列出的用户之外，所有其他用户都允许访问 crontab。如果既不存在 **cron.allow** 文件，也不存在 **cron.deny** 文件的话，那么在大多数系统上，就只有 root 用户才能提交 crontab（Debian 和 Ubuntu 默认允许所有用户提交 crontab）。

重要的是，应该注意，访问控制是由 **crontab** 而不是 **cron** 来实现的。如果某个用户能够通过其他方式把 crontab 文件秘密地转移到合适的目录中去，那么 **cron** 就会盲目地执行这个文件包含的命令。

8.4 cron 的常见用途

有一些标准任务特别适合由 cron 调用，这些任务通常组成了 root 用户的 crontab 文件的主体。在这一小节中，我们将介绍多种这样的任务以及实现它们的 crontab 行。

Linux 系统经常会带有一些预先安装好的 crontab 项，这些项主要是在**/etc/cron.d** 目录下。如果想使这些标准的配置项不起作用，那么可以在每行的开头插入一个“#”号，把它们注释掉。不要完全删除它们，也许以后还用得着。

除了采用**/etc/cron.d** 目录这种机制之外，Linux 发行版本还预先安装了若干 crontab 配置项，它们能运行在一组为大家所熟悉的目录中的脚本，从而给软件包提供了另一种不必对 crontab 文件做任何编辑，就能安装周期性任务的方法。例如，**/etc/cron.daily** 中的脚本每天运行一次，**/etc/cron.weekly** 中的脚本每周运行一次。

许多站点遇到了轻微但又重复出现的网络波动，因为系统管理员配置 cron 正好在同一时间在数百台计算机上运行相同的命令。采用 NTP 同步时钟的话会让问题恶化。对脚本或者配置文件调整，加一个随机的延迟时间，就很容易解决这个文件，但是要诊断出这个原因却挺棘手，因为症状来得快去得也快。

8.4.1 清理文件系统

有些文件在任何 Linux 系统中都是毫无价值的垃圾（并不是指系统文件）。例如，只要某个程序崩溃了，内核就会写出一个包含了该程序地址空间映像的文件 **core**[4]。core 文件对于软件开发人员而言是非常有用的，但对于管理员而言，它们通常只是一种对空间的浪费。用户通常对 core 文件一无所知，所以他们一般不会自己删除这些文件。

NFS 是那些多余文件的另一个来源。因为 NFS 服务器是无状态的，所以它们不得不使用一项特

[4] “core”的意思是“内存（memory）”。这个术语起源于早期的计算机系统，它们把装在网状结构上的很小的铁氧体磁芯作为内存单元。

www.TopSage.com

殊的规定来保存那些在本地计算机上已经被删除了但仍在由远程计算机使用的文件。多数实现是将这样的文件重新命名为**.nfs***xxx*，这里 *xxx* 是一个数字。许多不同的情况都会导致这些文件被遗忘，人们认为它们已经被删除了，而实际上它们却还留在系统中。在第 16 章中还将对 NFS 进行介绍。

很多程序会在**/tmp** 或者**/var/tmp** 中创建临时文件，因为这样或者那样的原因，这些文件没有被删除。有些程序，尤其是编辑器，喜欢为它们所处理的每个文件制作备份副本。

能部分解决这种垃圾文件问题的一个办法就是，用 **cron** 构造一些在夜间回收磁盘空间的工作。现代的系统一般带有这种功能，但我们最好还是检验一下自己系统的默认行为，确保它适用于您所处的情形。

下面是采用 **find** 命令的几个惯用法。

```
find / -xdev -type f '(' -name core -o name 'core.[0-9]*' ')' -atime +7
     -exec rm -f { } ';'
```

这条命令将删除那些在一个星期中都没有被访问的 core 映像文件。**-xdev** 参数用来确保 **find** 命令不会执行到根文件系统之外的文件系统中去，这项限制在那些交叉安装了很多文件系统的网络中是非常重要的。如果想要清理多个文件系统，那么可以对每个文件系统使用一条单独的命令（请注意，**/var** 通常也是一个独立的文件系统）。

系统管理员可以通过**/proc/sys/kernel/core_pattern** 文件设置给 core 文件起名字的方法。上面的 **find** 命令只处理 core 文件的默认名字（**core** 或者 **core.***PID*）。参数**-type f** 很重要，因为 Linux 的内核源代码包括一个目录也叫做 **core**，不应该删除它。

```
find / -xdev -atime +3 '(' -name '#*' -o -name '.#*' -o -name '*.CKP' -o
     -name '*~' -o -name '.nfs*' ')' -exec rm -f { } ';'
```

这条命令将删除那些用**#**、**.#**或者**.nfs** 开头的，或者用~以及**.CKP** 结尾且 3 天内都没有被访问过的文件。不同种类的临时文件和编辑器备份文件都是这一模式的典型代表。

出于性能方面的原因，有些系统管理员在安装文件系统的时候使用 noatime 选项，不让文件系统有访问时间戳。这项配置会让前面两条 **find** 命令发生误会，因为文件即使最近是活动的，但看上去都好像没有在用一样。遗憾的是，这种失灵的模式会删除文件，在使用上述命令之前，要保证文件系统带有访问时间。

```
cd /tmp; find . ! -name . ! -name lost+found -type d -mtime +3
     -exec /bin/rm -rf { } ';'
```

这条命令将递归删除**/tmp** 下在 72 小时之内没有被修改过的所有子目录。**/tmp** 中的普通文件会在系统启动时被系统启动脚本删除，但有些系统不会删除目录。如果存在名为 **lost+found** 的目录，那么就要特殊对待而不能删除它。如果**/tmp** 是一个独立的文件系统的话，这就非常重要了。请参考 7.6 节中的内容以获得关于 **lost+found** 的更多信息。

如果使用这些命令中的任何一个，都应该保证让用户知道您的清理策略。

8.4.2 配置文件的网络分布

如果您正在运行一个计算机网络，那么保留一个独立的、全网范围的、像邮件别名数据库（通常情况下是在**/etc/mail/aliases** 中）那样的配置文件版本往往会很方便。每天晚上可以使用 **rsync**、**rdist** 或者 **expect** 脚本来分发这些文件的主控版本。请参看第 17 章中的内容以获得关于共享配置文件的更多信息。

www.TopSage.com

有时候需要进行事后处理。例如，如果在 **sendmail.cf** 文件中没有设置选项 AutoRebuildAliases，那么可能要运行 **newaliases** 命令将一个邮件别名的文本文件转换为 **sendmail** 使用的散列格式。可能还需要将这些文件载入到如 NIS 这样的管理数据库中。

8.4.3 循环日志文件

Linux 能够很好地管理它的大多数日志文件，但是有些文件，除非手工对它们重新设置，否则就会无限膨胀。有很多方法可以用来防止日志文件溢出，最简单的方法就是定期截短它们。

一种更保守的方法就是通过保存每个日志的几个更老的版本来“循环”使用日志文件。这个方案可以防止日志文件失控，但又决不会让您遗漏任何近期的日志信息。因为日志的循环使用是周期性的而且是经常性的事件，所以它是一个理想的 **cron** 任务。请参看 10.1.2 节中的内容以获得更多信息。

8.5 其他的日程安排程序：ANACRON 和 FCRON

一般而言，**cron** 不能弥补因为系统宕机或者机器时间出现不连续的情况而错过执行的命令。不过，Vixie-**cron** 的确在努力，对于每隔一个小时以上的时间才执行一次的任务来说，如果时间的变化不到 3 个小时，仍然争取让任务正确执行。对于在时间上的小调整，比如像美国夏时制这样的情况，它能处理得不错。

膝上机以及其他并不长期开机的机器与 cron 的关系更不好，采用 cron 的候补软件对它们有好处。**anacron** 这个软件对 **cron** 有很好的补充作用，它是一种按照时间间隔来安排任务执行的工具。如果在系统的默认安装里没有包括 **anacron**，可以通过系统默认的软件包管理系统来安装它。

在 **anacron** 这个软件的世界里，用户只需要求某条特殊的命令每星期执行一次，但用不着指定它在每周一的凌晨 2 点执行。**anacron** 和 **cron** 不同，它维护着每条命令最近执行时间的记录，所以它只要比较这个时间、指定的时间间隔，以及当前时间就能判定是否需要再次执行命令。

anacron 提供了几种功能，有助于避免新启动的计算机对同时维护的大量任务应接不暇的问题。启动每项任务的延迟时间都不同，**anacron** 将命令逐条运行，所以一次只有一条命令激活。

对于系统管理任务来说，按时间间隔来安排执行，要比安排固定时间执行的方式更有用。遗憾的是，**anacron** 要求它的时间间隔按天来表示，这个事实限制了 **anacron** 作为一种一线的日程安排执行工具的功用。

anacron 本身必须在 **cron** 里使用。虽然 **anacron** 安排时间的粒度是天，但是每天多运行 **anacron** 几次就会有意义——如果知道 **cron** 始终能够在每天同一时间运行 **anacron**，那么可能就不会首先考虑使用 **anacron** 了。类似地，在系统启动的时候运行 **anacron** 也会有意义。

fcron 是一种更有雄心弥补 **cron** 不足的软件，它也包括类似 **anacron** 的功能，参见 fcron.free.fr。**fcron** 和 **anacron** 不同，它的目标明显就是要代替 Vixie-**cron**。它支持几种高级的日程安排方法，用 **cron** 和 **anacron** 的配置语言都做不到。不过，人们不得不质疑，比起 **fcron** 增加的复杂性和使用不标准的日程安排造成的混乱，这些功能的实用价值是否值得。

8.6 习题

E8.1 在您的系统上，一个本地用户以有规律的时间间隔运行开销很大的任务，滥用他的 crontab 特权。在几次要求他停止之后，您被迫收回他的特权。列出删除他的当前

www.TopSage.com

crontab 文件，并确保他不能增加新 crontab 文件所需的步骤。

E8.2 想出 3 项需要定期运行的任务（不包括本章提到过的那些任务）。写出每项任务的 crontab 配置项，指出配置项应该送入哪个 crontab 文件。

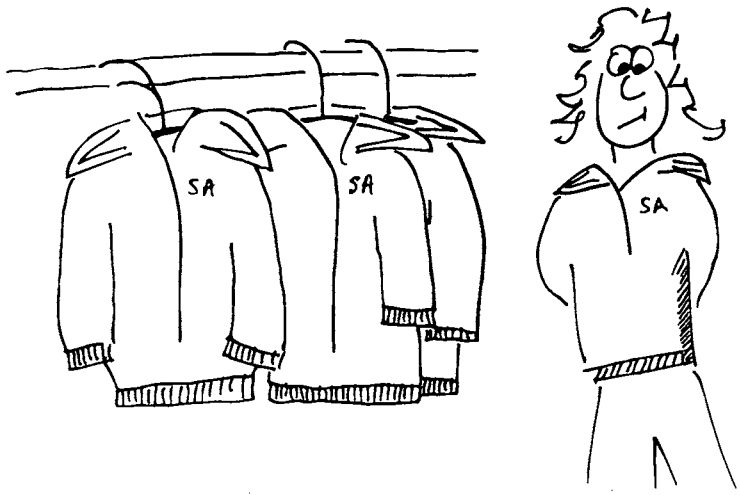
E8.3 找到您的 Linux 系统上的 crontab 文件，选出 3 个。解读每个文件，说明它何时运行、做什么、以及您认为为什么需要这个配置项（需要 root 权限）。

★E8.4 编写一个脚本，保持您有账号的所有机器上的启动文件（**~/.[a-z]***）同步。从 **cron** 定期运行这个脚本。（盲目地复制每个名字以点开头的文件安全吗？在目的主机上要被替换的文件被覆盖之前，应该做个备份吗？）

★E8.5 以 **du**、**sort** 和 **head** 命令的手册页作为参考，编写一个脚本，确定系统上最大的 10 个主（home）目录。在每周一的夜里 12:00 执行这个脚本，并且让它把输出结果发给您。提示：要用到按数字逆序排序（需要 root 权限）。

www.TopSage.com

第9章 备份



在大多数站点中，存储在计算机中的信息要比计算机本身更有价值。这些信息也很难替代。保护这些信息是系统管理员最重要的（而且很遗憾，也是最单调乏味的）任务之一。

丢失数据的方式成百上千，这些方式有的很有创造性，而有的则“不那么有创造性”。照例，软件缺陷（bug）会损坏数据文件。用户会意外地删除他们毕生的劳动成果。黑客以及那些对公司不满的雇员会删除磁盘上的数据。硬件问题和自然灾害则会破坏掉整个机房。

如果执行过正确的备份，那么就能让管理员把文件系统（或者是文件系统的任何部分）恢复成它上次备份时的状态。备份必须仔细进行，并且应该有严格的时间表。备份系统和备份介质也必须定期进行检查以确保它们能正确工作。

备份介质的完整性直接影响着公司的基本运营。高级管理人员需要理解备份实际上应该做什么，而不是他们想让备份做什么。如果某个大学计算机科学系丢失了一天的工作数据，可能没有太大影响，但如果是一个日用品贸易公司，就不会没有影响了。

我们将以一些备份的基本思想来开始本章的内容，随后讨论最常用的备份设备和备份介质（它们的优缺点以及成本等）。接下来，我们将讨论 Linux 的备份和存档命令，并且就哪种情况下最好使用哪个命令给出一些建议。然后，我们将讨论如何设计备份方案，并回顾一下流行的 **dump** 和 **restore** 工具的使用技巧。

最后，我们将看一看 Bacula，它是一个免费的网络备份软件包，我们还要对它的商用替代产品提一些看法。

www.TopSage.com

9.1 备份基本原理

在我们深入介绍备份之前，我们想讲述一些我们不断学到的普遍经验（通常是以艰难的方式学到的）。在这些建议中，没有任何一条经验是绝对的规则，但您会发现，您遵循的规则越多，进行备份的过程就会越顺利。

9.1.1 从一台机器执行所有的转储

许多备份工具都允许通过网络执行转储（dump）。尽管这么做会有一些性能上的损失，但由此带来的管理上的轻松却使之值得去做。我们已经发现的最好方法就是，在一个中心位置上运行一个脚本，（通过 **rsh** 或者 **ssh** 的方式）在需要转储的每台机器上执行 **rdump**，或者使用一个能够自动完成这个过程的软件包（商用的、免费的或者共享的软件）。所有转储都应该保存到同一个备份设备中（当然这个备份设备是非自动倒带的设备）。

如果网络太大，不能用一个单独的设备进行备份，也应该让备份系统尽可能集中。集中可以使管理更容易，还可以核实备份的完整性。根据所用备份介质的不同，往往可以在一台服务器上配置一台以上的备份介质设备而不会影响性能。

9.1.2 给备份介质加卷标

清楚而完整地给每个备份设备添加卷标是非常重要的。没有卷标的磁带是空磁带。应该在带身上做标记，唯一地标识出它的内容。诸如文件系统的列表和转储日期这样的详细信息都可以写在盒子上。

您必须能做到，不看转储脚本，就可以恢复根和**/usr** 文件系统。给文件系统的备份介质添加卷标时，可以用它们的格式、创建它们的 dump 命令的准确语法、以及不参考在线文档来恢复它们时所需的任何其他信息。

现在有一些免费的或者是商用的给备份添加卷标的程序。为了解决这个令您头痛的大问题，您可以考虑买一种这样的软件。如果您为自己的激光打印机购买了标签，那么通常情况下标签厂商会提供生成这些标签的（Windows）软件。更好的做法是，购买一种专用的 Windows 标签打印机。它们价格不贵，而且效果好。

9.1.3 选择合理的备份间隔

备份的频率越高，在系统崩溃时可能丢失的数据量就会越少。然而，备份会占用系统资源和操作者的时间。系统管理员必须以合理的时间和材料消耗为代价，提供充分的数据完整性。

在那些繁忙的系统中，通常情况下，每个工作日对主（home）目录中的文件系统进行备份是比较合适的。在那些使用不很频繁的系统或者那些数据变化不大的系统中，可能一周内做几次备份就足够了。在只有一个用户的小型系统中，一个星期做一次备份可能就够了。您的用户愿意丢失多少数据呢？

9.1.4 仔细选择文件系统

那些很少进行修改的文件系统不必像用户的主目录那样频繁地备份。如果在某个静态文件系统中只有少数文件有变化（比如说根文件系统中的**/etc/passwd**），那么每天都应该把这些文件复制到另外一

www.TopSage.com

个定期做备份的分区中。

如果**/tmp** 是一个独立的文件系统，那么就不应该备份它。**/tmp** 目录不应该包含任何重要的信息，所以没有任何理由去保存它。尽管这是很明显的，但据我们所知，有一个大型站点每天都对**/tmp** 进行备份。

9.1.5 在一卷磁带上做日常转储

在一种完美的情况下，您可以把自己的所有用户文件系统每天转储到一卷磁带上。对于某些站点而言，像 DLT、AIT 和 LTO 这样的高密度介质就可以完成。您可以每天在下班之前安装上一卷磁带，然后在晚上让 **cron** 替您做备份。这样，转储操作就会在文件不太可能发生变化的时候进行，并且转储操作对用户造成的影响也是最小的。请参考第 8 章中的内容了解关于 **cron** 的更多信息。

随着我们工作习惯的改变以及远程办公方式变得更为流行，适合做备份的时间范围正在缩小。根据我们的日志，我们看到凌晨 3：00 到 6：00 是最好的时间。不过，在 3 个小时候内备份 80TB 的数据不太可能。另一个主要问题是因为硬盘价格的降低造成计算机磁盘空间迅速膨胀。用户不再会购买磁盘空间不到 100GB 的台式机。当您只需要花上很少的钱就能够解决这个问题的时候，为什么还要对硬盘进行清理以及强行使用磁盘配额呢？遗憾的是，让计算机的联机存储容量超过备份能力所及太容易了。

如果一卷磁带装不下您的日常备份，那么还有几种选择：

- 购买一个更大容量的备份设备；
- 购买一台自动换带机或者磁带库，把多个存储介质安装到一台设备中去；
- 改变转储顺序；
- 编写更智能的脚本；
- 使用多个备份设备。

自动转储系统应该始终记录下它转储的每个文件系统的名称。保存好记录可以让您在需要恢复文件时能够迅速向前跳到正确的文件系统。在磁带的外部记录文件系统的顺序也是不错的想法（前面已提到过，但应当经常重申这个问题：一定要使用非自动倒带的磁带设备来把多次转储写入到磁带中）。

9.1.6 使文件系统小于转储设备

dump 以及其他容易得到的工具都能够完美地胜任将文件系统转储到多卷磁带中的操作。但如果一次转储跨越了多卷磁带，那么操作者必须待在现场来更换磁带，并且还应该对磁带仔细标记，使将来的恢复过程能够很容易地完成。除非您有充分的理由要去创建一个确实很大的文件系统，否则不要那么做。

9.1.7 异地保存磁带

多数机构都是把备份异地保存，这样一来，诸如失火这样的灾难就不会同时破坏原始数据和备份数据。异地保存的位置可以是任何地方，从银行里的保险箱，到总裁或者 CEO 的家里都行。那些专营安全保存备份介质的公司，能够为您的存档提供一个既安全又有空调的环境。一定要确保异地存储服务的提供商信誉好、有担保并且保险。现如今还有在线（但是异地）公司专营数据保护业务。

将磁带移至异地的速度应该取决于需要恢复文件的频率以及所能接受的等待时间的多少。一些站

www.TopSage.com

点通过每天进行两次转储的方式来避开非做决策不可的情形（这两次是转储到不同的磁带设备中）：一份转储放置在本地，另外一份转储被立刻移至异地[1]。

9.1.8 保护备份

安全顾问 Dan Geer 说，“备份会做些什么呢？它毫无疑问会在异地使文件权限受到侵害。”

做好备份磁带的安全工作。因为它们包含了单位的所有数据，任何能在物理上接触到备份的人都能读取它们。不仅应该把那些磁带保存在异地，还应该妥善保存它们。如果您为了达到这个目的而使用了商用存储设施，那么与您打交道的那个公司应该保证磁带在其关照之下的机密性。

对备份磁带加密是一种可以考虑的方法。许多商业备份工具都能让加密相当轻松。另一方面，必须保证加密密钥不会丢失或者破坏，而且在紧急时刻能拿到。

有些公司非常看重备份的重要性，所以做了双备份，实际上这的确是个不错的主意。

9.1.9 执行转储期间限制活动

在转储过程中应该对文件系统的活动进行限制，因为那些变化会引起备份工具出错。可以通过在只有很少的活动用户时（半夜或者周末的时候）进行转储操作，或者通过使文件系统只能由备份工具访问的方式来限制活动（后一种预防措施从理论上讲还不错，但却很少能实现。用户想要对所有文件系统具有 24/7 的访问能力。如今已不可能在没有磁盘活动的情况下做备份了）。

类似 Network Appliance 公司制造的专用文件服务器，可以通过定期对文件系统做“快照（snapshot）”来提供在线备份，而且这个时间间隔是可调的。这项功能可以对活动的文件系统进行安全备份，这也是使用专用文件服务器的重要优点之一。请参看 16.5 节了解关于文件服务器的更多信息。

9.1.10 检查磁带

我们已经听说过很多关于系统管理员的可怕故事，这些系统管理员直到发生了一起严重的系统错误之后才发现转储工作的问题。关键在于，您应该连续监视备份过程，并且验证它在正确工作。操作人员的错误对转储的破坏比任何其他问题都要多。

第一项检查就是在完成转储之后让转储软件马上试着去重新读取这些磁带[2]。扫描磁带以验证它包含了期望数目的文件，就是一次良好的检查。如果能够扫描每卷磁带，那就最好了。但对于每天使用数百卷磁带的大型机构而言，这种方法就不可行。在这种环境下，最谨慎的做法是随机抽查。

为每个文件系统生成一个目录（**dump** 命令的用户可以使用 **restore -t** 命令），并且把结果存储在磁盘上，这往往是有用的。这些编目的名字应该和对应的磁带相关，比如说，**okra:usr.Jan.13**。有了一个星期的记录，就可以很容易地找到某个丢失的文件在哪卷磁带上。您只需使用 **grep** 命令检索要找的文件名，然后挑出最新的一个实例即可。参考 9.4 节了解有关 **restore** 的更多信息。

除了提供磁带的编目之外，成功从磁盘读取目录，可以表明转储结果良好，在您需要的时候就能

[1] 位于世贸中心的一家大型金融机构把它的“异地”备份放在办公室楼下一两层的地方。当这座建筑（第一次）被撞的时候，备份磁带（以及计算机）都毁掉了。要保证真正做到了“异地”。

[2] 您可以用 **restore -C** 来校验转储磁带和目录树。

www.TopSage.com

够读取该磁带。迅速尝试一下恢复任意一个文件将会增强您对自己能从磁带中恢复文件系统的自信[3]。

您还应该定期尝试从不同的磁带恢复备份，以确认备份仍能恢复。应该时常试着从旧转储磁带（几个月或者几年之前的磁带）中进行恢复操作。随着时间的推移驱动器的磁头会无法和磁道对齐，从而不能读取那些旧磁带。这些磁带的数据可以通过专门提供这种服务的公司来进行恢复，但这非常昂贵。

还有一项相关检查，就是验证您不只能在自己的硬件上读取这些磁带。如果机房失火，那么即使知道转储磁带能够被某个已经毁坏的磁带驱动器读取也是没有多大用处的。以前，DAT 磁带特别容易发生这种问题，但是这一技术最近的版本已经得到改进。

9.1.11 发掘磁带的寿命

磁带只有有限的使用寿命。能循环使用存储介质当然很好，但一定要遵守制造商们关于磁带寿命的建议。大多数制造商是根据磁带能够承受的周转次数来对磁带的寿命进行量化的：一次备份、一次恢复以及一次 **mt fsf**（文件向前跳进）操作都代表一次磁带周转。非磁带技术的寿命则长得多，有时候用平均故障时间（MTTF，mean-time-to-failure）来表示。

在把老磁带扔进垃圾箱之前，要记得清除磁带或者让它们不可读取。批量磁带清除机（一种大型电磁设备）有助于完成这项工作，但要确保让它远离计算机和正在用的磁带。剪断或者拽出一盘备份磁带的一部分都不能真正保护数据，因为很容易接合断开的磁带或者把磁带重新绕回去。付费的话就能让文件销毁公司把磁带切碎。

9.1.12 为备份而设计数据

随着硬盘变得那么便宜，而且新的存储体系结构又那么可靠，不对所有数据进行备份成为诱人的目标。有了一种合理的存储体系结构——它是随着磁盘需要的增加而设计出来的，而不是随意出现的——很容易就能完成大量备份工作。您可以从制作一份自己的存储需求清单开始：

- 站点要处理的不同类型的数据；
- 每种类型的数据预期的变更率；
- 为避免可能丢失数据而确定的备份频率；
- 数据会散布的行政范围。

您可以使用这个信息，在考虑到备份和潜在增长的条件下设计自己站点的存储体系结构。将项目目录和用户主目录保存在一个专用的文件服务器中可以使数据管理以及数据安全的保障更容易一些。

9.1.13 作最坏的准备

在建立起一个备份过程之后，应该研究一下可能出现的最坏情形：您的站点被完全破坏了。确定一下有多少数据会丢失，以及需要花上多长时间才能让自己的系统重新开始工作（包括花在购买新硬件上的时间）。然后决定自己能否承受这个后果。

9.2 备份设备和介质

因为很多类型的故障都可以一次破坏好几个硬件设备，所以备份应该写入到某种可移动的介质中

[3] 例如，**restore-t** 命令只读取转储磁带的目录，此目录保存在磁带的开头。在真正开始恢复一个文件时，则相当于在测试该介质更大范围的区域。

www.TopSage.com

去。例如，将一个硬盘中的数据备份到另外一个硬盘中（尽管这比完全不进行备份要好一些）几乎不能保护不受硬盘控制器故障的影响。在过去的几年中，通过 Internet 对数据进行备份的那些公司已经进入到了这种移动存储的时代，但大多数备份仍然是在本地保存。

很多类型的介质都是使用磁性颗粒来保存数据的。这些介质容易受到电场或者磁场的破坏。下面是要避免的一些特殊危险：音频扬声器、变压器和电源、没有外壳的电机、硬盘风扇、CRT 监视器、甚至长期暴露在地球的背景辐射。所有磁带经过若干年后都会变得不可读。大多数介质可以保存 3 年，但如果您打算把数据保存得更长一些，那么就应该使用别的介质或者重新把数据记录在磁带上。

下面的几小节将描述一些用于备份的介质。对这些介质的介绍大体上是根据容量从小到大的顺序来进行的。

制造商喜欢根据压缩后的数据来提供磁带的容量，他们常常会乐观地假定 2:1 或者更高的数据压缩比率。在下面几节中，我们会忽略掉压缩，而只提出能够物理存储在每个存储介质中的实际字节数。

假定的压缩比率还会影响驱动器的吞吐速率。如果一个驱动器在物理意义上能够以 1MB/s 的速率写入到磁带中，但制造商假定了 2:1 的压缩比率，那么吞吐量就会神奇地上升到 2MB/s。和磁带容量的表示一样，下面我们也忽略掉了吞吐量的夸大。

尽管成本和介质容量都是重要的考虑因素，但考虑吞吐量也很重要。人们更乐于使用速度快的介质，而且使用速度快的介质在转储的安排上也更灵活。

在本书的上一版里，我们提到了传统和超级容量（例如，Zip 盘）两种软盘，把它们也当作备份介质。这些介质发展都很成熟，但是由于您不大容易再买到带有软驱的膝上机，所以我们现在把这两种介质从我们的清单中剔除。虽然软盘对于交换数据来说很有用，但是它们的成本太高，选用它们做备份很不合适。

9.2.1 光盘：CD-R/RW、DVD±R/RW 和 DVD-RAM

CD 和 DVD 的成本只有大约每张 0.40 美元，对于小型独立系统的备份而言，是一种很有吸引力的选择。CD 的容量大约为 650MB，DVD 的容量为 4.7GB。正在成为主流的双层 DVD 容量有 8.5GB。

每种常见的总线（SCSI、IDE 和 USB 等）都有能写这类介质的驱动器，而且在许多情况下，这些设备都便宜到近乎于不要钱的程度。CD 光盘过去也大量使用，但是现在似乎已经过气了。既然 CD 和 DVD 的价格相称，所以除了现实中 CD 的光驱仍然比较常见之外，没有理由不用 DVD 而用 CD。

光学介质是通过在光化学过程中使用一束激光来写数据的。虽然“硬”保存的数据寿命还不好说，但是人们普遍认为光学介质肯定要比磁性介质的寿命长。不过，一次性刻录的光学介质（CD-R、DVD-R 和 DVD+R）没有制成品（压制）的 CD 和 DVD 经久耐用。

CD-R 不合适用于常规备份，但适合做未来长时间以后恢复的数据存档。CD-RW 适合做定期小批量个人数据的存档。

现如今，快速 DVD 刻录机的速度如果不比磁带机块，也已经和它一样快了。DVD 的一次性刻录版本是 DVD-R（Pioneer 公司在 1997 年的研发成果）和 DVD+R（多家公司在 2002 年联合研发的成果）。DVD-RW、DVD+RW 和 DVD-RAM 都可以刻录。DVD-RAM 系统内建纠错管理功能，因此比其他光学存储介质更可靠。另一方面，它的价格也更贵。

据制造商估计，如果这些光学介质得到妥善保存，那么它们的预计寿命可达数百年。制造商建议的正确保存方法包括采用单独的外套，保存在一个温度恒定在 41～68 华氏度之间，相对湿度在 30%～50%之间，不会受到阳光直射的环境里，并且用水溶性记号笔标记。在平均条件下，有 1～5 年的可靠保质期可能更实际一些。

根据多种第三方的评估，光学介质的可靠性经证实特别依赖于制造商。购买高质量的介质就能有好效果。遗憾的是，甚至在一家制造商的生产线上，产品之间的质量也不一样，所以没有能打保票的

www.TopSage.com

制造商。近年来，Taiyo Yuden 公司在可靠性上力拔头筹，但也仅限于在日本制造的介质，Taiyo Yuden 在其他国家生产的介质良莠不齐。

Blu-Ray 光盘最近进入光学数据存储市场，它的各种形式能够存储 25～100GB 的数据。容量大是读写光盘采用了短波激光（405nm）的缘故（Blu-Ray 中的 Blue[蓝]由此得名）。它的第一种播放器于 2006 年 6 月在美国面世。这项技术值得关注，它有希望成为备份的良好解决方案。

9.2.2 可移动硬盘（USB 和火线）

通过 USB 或者火线（IEEE 1394）端口连接的外部存储设备已经变得很常见了。支撑它的存储技术通常是硬盘的某种形式，但是在低端（无处不在的“JumpDrive”）采用闪存设备的情况也很多。它的容量从不到 1MB 到 600GB 以上。目前闪存设备容量的上限大约为 4GB。

这些设备的主要局限性在于总线速率，除非采用了 USB 2.0 和 FireWire 800，两者的吞吐率都能达到可观的 50～90MB/s 这个范围。

闪存设备的寿命几乎就取决于写数据的次数。中等程度的设备应该能写数百万次。

9.2.3 小型磁带机：8 毫米磁带和 DDS/DAT

各种 8 毫米和 DDS/DAT 磁带机组成了低端磁盘存储市场。Exabytes 公司的 8mm 磁带机是早期的型号，但是这种磁带机每隔 6～12 个月磁头就无法对齐磁道，需要送修。这种磁带 2～7GB 的容量使得它们都不适合用于备份现如今的桌面系统，更不用说服务器了。

DDS/DAT（Digital Audio Tape，数字音频磁带）磁带机是使用 4mm 盒式磁带的螺旋扫描设备。虽然这类设备通常叫做 DAT 磁带机，但它们实际上是 DDS（Digital Data Storge，数字数据存储）磁带机。准确地区别它们并不重要。这种磁带一开始的格式容量为 2GB，但是后几代产品大大提高了 DDS 的容量。目前这一代产品（DAT 72）的数据容量达到 32GB，传输速率为 3.5MB/s。这种磁带应该经得起 100 次备份操作，根据报告，它们的寿命有 10 年。

9.2.4 DLT 和 S-DLT

DLT/S-DLT（Digital Linear Tape，数字线性磁带/Super Digital Linear Tape，超级数字线性磁带）是一种主流的备份介质。这类磁带机性能可靠，价格不高，并且可以保存大量数据。它们源自 DEC 的 TK-50 和 TK-70 盒式磁带机。DEC 将这项技术卖给了 Quantum 公司，Quantum 公司通过提高它们的速度和容量以及降低它们的价格使之得到了普及。2002 年，Quantum 公司获得了 Benchmark Storage Innovations 公司的 Super DLT 技术，这项技术让记录磁头前后倾斜，减少了相邻磁道之间的串扰。

Quantum 公司现在提供两种硬件产品线：高性能产品线和高性价产品线。用户花什么样的价钱就能买到什么样的产品。磁带容量从 DLT-4 的 800GB 到 DLT-4 高性价产品线的 160GB，两者的传输速率分别为 60MB/s 和 10MB/s。制造商宣传这种磁带的寿命有 20～30 年——也就是说，如果能读它们的硬件还存在，这种磁带就能用。可现如今还有多少 9 道磁带机仍然可以工作还能用呢？

S-DLT 的缺点是磁带的价格高，每盘大约 30～45 美元。对于大学来说昂贵了点，或许对于华尔街的投资公司来说并不贵。

9.2.5 AIT 和 SAIT

AIT（Advanced Intelligent Tape，先进智能磁带）是 Sony 公司自己的 8mm 产品。1996 年，Sony

www.TopSage.com

公司解除了它和 Exabyte 公司的关系，并推出了 AIT-1，这是一种 8mm 螺旋扫描设备，其容量是 Exabyte 公司 8mm 磁带容量的两倍。自从最初产品发布以来，Sony 公司已经推出了一种 AIT-1 的高容量版本以及后续的 AIT-4，AIT-4 的容量达到 200GB，最高传输速率为 24MB/s。

SAIT 是索尼的半高产品，它使用的磁带比 AIT 大，容量也比 AIT 大。SAIT 磁带能够保存 500GB 的数据，传输速率达到 30MB/s。这种磁带以磁带库产品的形式使用最为流行。

AIT 磁带机中使用的 AME（Advanced Metal Evaporated，高级金属薄膜）磁带具有很长的使用寿命。它们还包含一个内置的 EEPROM，能让介质本身有一些智能。但是，任何 EEPROM 的实际使用都需要有软件支持。这种磁带机和磁带的价格和 DLT 大致相同。

9.2.6 VXA 和 VXA-X

Exabyte 公司现在的产品包括 VXA 和 VXA-X 技术。VXA 驱动器使用 Exabyte 公司称之为分组数据传输的技术。VXA-X 产品仍然依赖 Sony 公司的 AME 磁带，V 系列产品随着更大容量的磁带出现可以随之升级。VXA 和 X 系列的产品标称容量在 33～160GB，传输速率为 24MB/s。

9.2.7 LTO

IBM、HP 和 Quantum 公司共同开发了 LTO（线性磁带开放标准，Linear Tape-Open）用做专有 DLT 格式的替代格式。最新版的 LTO-3 容量为 400GB，速度达到 80MB/s。LTO 磁带的预计存储寿命为 30 年，但是容易受到漏磁的影响。400GB 磁带的成本大约为 80 美元。

9.2.8 自动选带机、自动换带机以及磁带库

现在随着低成本硬盘的使用，大多数站点的硬盘空间都相当大，以至于一次完整的备份即使在每卷磁带存储 100GB 数据的情况下，也需要多卷磁带才能完成。对于这些站点而言，一种可能的解决方案就是使用自动换带机（stacker）、自动选带机（jukebox）或者磁带库（译者注：jukebox 和 stacker 最大的区别就在于，jukebox 能够以任意次序选取磁带，因此较为智能。而 stacker 则只能依次更换磁带，所以把它们分别称为自动选带机和自动换带机）。

自动换带机是和标准磁带机一起使用的一种简单的更换磁带设备。它具有可装载磁带的送带箱。当那些已写满的磁带从驱动器中弹出时，它会退出磁带，然后用送带箱中的空白磁带替换它们。大多数自动换带机都能存放大约 10 卷磁带。

自动选带机是一种能够在一定数目的磁带机中自动调换可移动磁带的硬件设备，这和一种老式的能够在一个转盘上更换唱片的自动点唱机很相似。这里讨论的所有磁带都可以使用自动选带机。自动选带机经常和能够理解如何操纵更换机的专用备份软件捆绑销售。Storage Technology 公司（现在的 Sun 微系统公司）和 Sony 公司是生产这类产品的两个制造商。

磁带库是一种用于大型数据集合——它们通常达到太字节（TeraBytes，10^{12} 字节）——的硬件备份解决方案。它们是带有多卷磁带（或者 CD）机的厨柜大小的机械装置，还有一个用来在磁带库的很多层架子上对介质进行检索和归档的机械臂。可以想象，购买和维护它们是非常昂贵的，而且它们要求有专用的电源、空间以及空调。大多数磁带库的购买者还需要购买一份来自制造商的运行合同来优化和运行设备。当然，磁带库还有一个软件部件，它在设备上实际运行。Storage Technology 公司（Sun 微系统公司）是磁带库制造商中的领头羊。

www.TopSage.com

9.2.9 硬盘

不断降低的硬盘驱动器成本成为考虑从硬盘到硬盘做备份的一个吸引人的原因。尽管我们建议不要在同一台机器上将一个硬盘上的数据复制到另外一个硬盘里去，但在网络上，硬盘是一个低成本的好解决方案。

一个明显的问题就是硬盘的存储空间是有限的，并且最终必须重新使用。但是，从硬盘到硬盘的备份是防止文件被意外删除的极好办法。如果通过 NFS 或 CIFS 共享，在大家熟知的位置上保留一个每天更新的硬盘映像，那么用户无需管理员的帮助就能够进行错误恢复。另一种流行的选择是采用如今的大容量火线和 U 盘做台式机的备份。Cintre ZDisc 1-Button Instant Backup 是一种 3.5 英寸的外置 USB 2.0 硬盘，这种富有吸引力的设备价格大约在 100 美元。

9.2.10 介质类型小结

介质的类型太多啦！表 9.1 对上面各小节中讨论到的介质的特性进行了汇总。

表 9.1 备份介质比较

介　质	容量[a]	速度[a]	驱动器	介质	每 GB 成本	是否能够重用?	是否随机方式?[b]
CD-R	650MB	4 MB/s	$15	20¢	32¢	否	是
CD-RW	650MB	4 MB/s	$20	30¢	48¢	是	是
DVD±R	4.7GB	10 MB/s	$50	40¢	9¢	否	是
DVD+R DL[c]	8.5GB	10 MB/s	$50	$2	24¢	否	是
DVD±RW	4.7GB	10 MB/s	$50	80¢	17¢	是	是
DDS-4 (4mm)	20GB	10 MB/s	$300	$5	25¢	是	否
DLT/S-DLT	160GB	10 MB/s	$1 000	$20	13¢	是	否
AIT-4 (8mm)	200GB	6 MB/s	$2 500	$40	20¢	是	否
SAIT-1	500GB	30 MB/s	$6 500	$126	25¢	是	否
VXA-172	86GB	12 MB/s	$600	$30	35¢	是	否
VXA-320	160GB	12 MB/s	$800	$60	35¢	是	否
LTO-3	400GB	30 MB/s	$3 000	$65	16¢	是	否

a. 未经压缩的容量和速度。
b. 能随机访问介质的任意地方吗?
c. 双层。

W.Curtis Preston 已经编纂了一份按制造商归类的备份设备参考清单。

9.2.11 设备选型

当您要购置一套备份系统的时候，最好能把表 9.1 的内容搞得很清楚。所有种类的介质都能有比较好的表现，在那些价格比较接近的技术中，一般没有非选哪一种不可的理由。只要购置能够满足您的要求和预算的系统就行了。

对于小型工作组和存储量很大的个人计算机来说，DDS、AIT 和 LTO 驱动器是优秀的解决方案。它们的启动成本相当适中，磁带也到处都能买到，而且每种标准都有多家制造商采用。这几种系统的

www.TopSage.com

速度都足够快，可以在有限的时间里备份大量数据。

DLT、AIT 和 LTO-3 大致可比。在这三者之间没有一个很明显的赢家，即使有过，这一情形也毫无疑问将在几个月之内随着新版格式的发布而改变。所有这些格式都相当成熟，而且很容易集成到您的环境中，即大学和企业的环境。

在下面各小节中，我们将使用一个普通的名词“tape（磁带）”来指那些选作备份的介质。备份命令的例子用磁带设备的术语来表达。

9.3 用 dump 建立增量备份机制

dump 和 **restore** 命令是用来创建备份以及从备份恢复的最常见的方法。这两个程序已经存在很长时间了，它们的用法众所周知。在大多数站点中，**dump** 和 **restore** 都是支撑自动备份软件使用的命令。

根据您在最初安装 Linux 时所作的选择的不同，您可能不得不在 Linux 系统上明确安装 **dump** 和 **restore** 命令。我们举例的系统都有软件包，很容易就可以安装好。当前的 Red Hat 和 Fedora Core 在安装时就提供给系统管理员一个包含 **dump** 的软件包。

在 Linux 下，没有什么是静态链接的，所以您需要**/lib** 下的共享库来做些有用的事情。静态链接能够让灾难恢复更容易，因为静态链接的 **restore** 自己包含了要链接的库。

9.3.1 转储文件系统

dump 命令用来建立一份自上次转储操作以来进行过修改的文件列表，然后把这些文件打包成一个单独的大型文件，在外部设备上存档。**dump** 命令和本章后面要讲到的大多数其他工具相比较而言，具有以下一些优点：

- 备份可以跨越多卷磁带；
- 任何类型的文件（甚至是设备）都能备份和恢复；
- 访问权限、归属关系和修改时间都被保存下来；
- 有“孔洞”的文件能够被正确处理[4]；
- 备份能够以增量的方式进行（只把那些最近修改过的文件写到磁带上去）。

Linux 上使用的 **tar** 是 GNU 的版本，它也能提供所有这些功能。不过，**dump** 对增量备份（后面讨论）的处理要比 **tar** 稍微高级一点儿。如果您的需求比较复杂，那么您就会发现这多出来的功能是有用处的。

实际上，在 Linux 环境下，非要选择 **dump** 而不是 **tar** 的理由和 Linux 根本没有什么关系。遗憾的是，大多数主流 UNIX 发行版本所附带的 **tar** 版本缺乏 GNU 的 **tar** 所具备的许多功能。如果您必须对 UNIX 和 Linux 都提供备份支持，那么 **dump** 是您最好的选择。它是唯一能够在不同平台上（相当）一致地处理好这些问题的命令，这样一来，您只要是一条命令的专家就够了，不必熟悉两条命令。如果您很幸运，处在一个纯 Linux 环境里，那么就可以选用自己最喜欢的工具啦。**dump** 的功能不足，而 **tar** 就好用多了！

dump 命令懂得原始文件系统的布局，它直接读取一个文件系统的 inode 表以决定哪些文件必须

[4] 孔洞是指从未包含任何数据的文件块。如果您打开一个文件，写入一个字节，在文件中向前搜索 1MB，然后再写入另外一个字节，这样即使文件的逻辑大小要大得多，但得到的“稀疏”文件只会占用两个磁盘块。用伯克利 DB 和 **ndbm** 命令创建的文件都会包含很多孔洞。

www.TopSage.com

备份。有关文件系统的这些知识使得 **dump** 非常有效，但它也带来了一些限制[5]。

第一条限制就是每个文件系统必须单独转储备份。如果您有一个已经分过区的硬盘，那么必须对每个分区单独进行转储备份。另一项限制就是只有本地计算机中的文件系统才能够进行转储。从一台远程计算机安装的 NFS 文件系统不能转储备份。但可以使用 **dump** 的配套命令 **rdump** 把本地文件系统转储到远程的磁带驱动器中。请参看第 17 章以获取关于 NFS 的更多信息。

dump 命令最重要的特性就是它支持“增量”备份这一概念。尽管有可能每天都对整个文件系统进行备份，但这在通常情况下并不切合实际。利用增量备份，可以只对那些从上次备份之后进行了修改的文件进行备份。

在做备份的时候，要指定一个备份级别，它是 0～9 之间的一个整数。级别为 N 的转储会对从上次进行的级别小于 N 的转储操作以来修改过的所有文件进行备份。级别为 0 的备份会把整个文件系统转储到磁带上。使用增量备份系统后，要把文件系统重新恢复到它在上次备份时所处的状态，就得从若干套备份磁带中恢复文件[6]。

dump 命令的另外一个好特性就是它不关心文件名的长度。目录层次可以任意深，还可以正确处理长文件名。

dump 命令的第一个参数必须是增量转储级别。**dump** 使用**/etc/dumpdates** 文件来决定增量转储必须倒回去多远。**-u** 标志可以使 **dump** 命令在转储完成之后自动更新**/etc/dumpdates** 文件，将日期、转储级别和文件系统的名称都记录下来。如果您从来没有指定过**-u** 标志，那么所有转储都会变为级别 0，因为那样的话，将不会有先前备份过当前文件系统的记录。如果要修改文件系统的名称，可以手工编辑**/etc/dumpdates** 文件。

dump 命令会把它的输出发送到某个默认设备上，通常情况下是主磁带机。如果想使用不同的设备，可以使用**-f** 标志来通知 **dump** 命令将其输出发送到别处。如果在一卷磁带上保存多次转储的数据，那么应该确保指定了一个不会自动倒带的磁带设备（关闭时不会引起倒带的一个设备文件——大多数磁带驱动器都有一个标准设备项和一个非自动倒带设备项）。请阅读磁带设备的手册页以确定相应设备文件的确切名称[7]。Linux 通常用**/dev/st0** 代表倒带设备，而用**/dev/nst0** 代表非倒带设备。请参考 28.2.1 节，了解关于设备号的更多信息。

如果您不小心选择了自动倒带设备，那么最终只能保存最后转储的那个文件系统。因为 **dump** 命令并不知道磁带定位在哪里，这个失误当时不会造成出错。这种情况只有在试图恢复文件的时候才会显现出来。

当使用 **rdump** 命令转储数据到一个远程系统时，必须把远程磁带驱动器指定为 *hosnamet:device* 的形式。例如：

```
# rdump -0u -f anchor:/dev/nst0 /spare
```

访问远程磁带机的权限由 SSH 通道控制。请参看 20.11.3 节，了解更多的信息。

以前，必须确切地告诉 **dump** 命令磁带有多长，这样它就会在用完磁带之前停止写入操作。如今，多数磁带驱动器能够在磁带到达末尾时做出判断，并把这一情况报告给 **dump** 命令，**dump** 命令随后就会自动倒带并弹出当前磁带，然后要求换入一卷新磁带。因为硬件压缩的不定性会使每卷磁带的“虚拟长度”有点难以确定，所以，如果硬件支持的话，最好是根据磁带的结束标志来进行判断。如果硬件不支持，那么您可以用**-B** 选项以 KB 为单位指定磁带的长度。

[5] **dump** 命令需要访问原始磁盘分区。允许执行转储的任何用户只要稍做一点工作，就能够读取系统上的所有文件。

[6] 实际上，大多数版本的 **dump** 并不会记录已删除的文件。如果您从增量备份中进行恢复，那么删除掉的文件也将会重新创建。

[7] 磁带单元的所有项都使用相同的主设备号。次设备号告诉驱动程序有关特殊行为的信息（自动倒带、字节顺序交换等）。

www.TopSage.com

```
# dump -5u -B 2000000 -f /dev/nst0 /work
DUMP: Date of this level 5 dump: Wed May  8 16:59:45 2006
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/hda2 (/work) to /dev/nst0
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 18750003 tape blocks on .23 tape(s)
....
```

-B 标志（磁带的长度：20GB）和**-f** 标志（磁带设备：**/dev/nst0**）前面的标志是**-5u**。最后一个参数必须要有，它是要转储的文件系统的名字（**/work**）。

9.3.2 转储顺序

因为转储备份的级别只有在相对其他级别而言时才有意义，所以转储备份可以按照各种安排来执行。适合于您的安排取决于：

- 文件系统的活动情况；
- 转储备份设备的容量；
- 所需的冗余量；
- 愿意购买的磁带数量。

在备份一个文件系统需要很多磁带的时期，复杂的备份顺序对于减少每天备份所消耗的磁带数量是有用的。随着磁带容量的增大，在转储级别之间进行严格的区分就没有多大用处了。

因为多数文件从来都不进行修改，所以，即使安排最简单的增量备份，也可以免去很多文件的日常转储备份。当您安排更多的转储级别时，就把那些原本已经相当少的活动文件分成了越来越小的片断。

复杂的转储安排提供了下面 3 项好处：

- 您可以更加频繁地备份数据，控制潜在的数据损失；
- 您每天可以使用更少的磁带（或者将所有数据都存储在一卷磁带中）；
- 您可以保存每个文件的多个副本，以防磁带出错。

一般情况下，选择转储顺序的方法就是确定您在这 3 个方面的需要。根据这些限制条件，您可以设计出一个复杂度适宜的安排方案。我们将讲述几种可能的转储顺序以及这些顺序背后的动机。它们中的某个顺序可能适合于您的站点——或者，您也可能需要一个完全不同的安排。

一、简单的安排

如果磁盘空间总量比磁带设备的容量小，那么可以完全直接转储。您可以每天对每个文件系统都进行 0 级转储。您可以重复使用一组磁带，但应该永久保存每隔 N 天（这里 N 是由站点需要所决定的）的一卷磁带。这种安排每年的花费为：

（365/N）×单卷磁带价格

不要在每晚备份时重复使用同一卷磁带。最好是循环使用一组磁带，这样即使某天晚上的转储备份报废了，您仍然能够后退到前一天晚上的备份。

这种安排提供了巨大的冗余度，并且使数据恢复非常容易。对于那些具有很多资金，但只有有限的操作时间（或技能）的站点而言，这是一个很好的解决方案。从安全和方便的角度来看，这种安排非常理想。如果没有特别原因（比如说，为了节省磁带和力气），就可以使用这种方法。

二、适中的安排

对于大多数站点而言，更为合理的安排就是为每个星期中的每一天、每个月中的每个星期（需要 5 个）以及每年中的每个月都分配一卷磁带。每一天向日磁带做 9 级转储备份。每个星期向周磁带做 5 级转储备份。每个月向月磁带做 3 级转储备份。只要转储信息增大得太多，以至于无法保存在一卷磁

www.TopSage.com

带中，就应该做一次 0 级转储备份，这在每月的磁带中最有可能发生。每年至少做一次 0 级转储备份。

级别 3、5 和 9 的选择是随意的。您也可以使用级别 1、2 和 3，效果一样。然而，如果以后决定添加另外一个转储级别时，转储级别之间的间隔将会给您一些喘息的空间。

这种安排需要 24 卷磁带，再加上用来完成级别为 0 的转储所需的磁带。尽管它并不需要太多磁带，但也不能提供多少冗余性。

9.4 用 restore 从转储中恢复

从转储磁带中提取数据的程序叫作 **restore**。我们首先讨论恢复单个文件（或一小组文件），然后说明如何恢复整个文件系统。

9.4.1 恢复单个文件

当您得知有一个文件丢失时，应该采取的第一个步骤就是确定哪卷磁带中包含有该文件的版本。用户常常需要文件最近的版本，但并不总是如此。例如，一个用户在无意之间把另外一个文件复制到这个文件上面，造成该文件丢失，他需要的是这次意外发生之前的文件版本。如果您能够让用户不仅告诉您丢失了什么文件，还告诉您这些文件是何时丢失的以及上次是何时修改的，这将非常有帮助。我们发现，用一张请求表格来组织好用户反映的情况很有帮助。

如果您没有在线目录，那么必须安装各个磁带，并重复尝试从中恢复丢失的文件，直到发现正确的磁带为止。如果用户记起了文件上次是何时进行修改的，那么您或许可以受到启发，猜测出文件在哪卷磁带。

在决定了需要从哪卷磁带中提取数据之后，就应该创建或者执行 **cd** 命令进入到诸如**/var/restore** 这样的可以创建一个大型的目录层次的临时目录。**restore** 的大多数版本都必须在文件被恢复之前，创建通向这个特定文件的所有目录。不要使用**/tmp**——如果在所恢复的数据移动到原始位置之前计算机崩溃或者重新启动，那么您的工作就会被删除。

restore 命令有很多选项。最有用的就是用于交互式恢复单个文件和目录的**-i** 选项，以及用于完全恢复整个文件系统的**-r** 选项。您可能还需要使用**-x** 选项，它请求非交互式地恢复特定文件——请注意不要覆盖已经存在的文件。

restore -i 从磁带中读取备份目录，然后让您通过使用 **ls**、**cd** 和 **pwd** 这样的命令像一个普通的目录树那样遍历转储目录。您可以使用 **add** 命令来标记那些需要恢复的文件。选好后，键入 **extract** 命令将文件从磁带中提取出来。

如果在一卷磁带中存放了多个文件，那么必须在运行 **restore** 命令之前使用 **mt** 命令把磁带定位在正确的转储文件处。请记住，一定要使用非自动倒带的设备。请参看 9.7 节以获取关于 **mt** 的介绍。

例如，为了从远程磁带驱动器中恢复文件**/users/janet/iamlost**，可能需要发出以下命令。假定您已经找到了正确的磁带，并且将其安装到了 **tapehost:/dev/nst0** 上，而且明确知道包含 janet 的主目录的文件系统是磁带上的第 4 个文件。

```
# mkdir /var/restore
# cd /var/restore
# ssh tapehost mt -f /dev/nst0 fsf 3
# rrestore -i -f tapehost:/dev/nst0
restore> ls
.:
janet/  garth/  lost+found/ lynda/
```

www.TopSage.com

```
restore> cd janet
restore> ls
afile bfile cfile iamlost
restore> add iamlost
restore> ls[8]
afile bfile cfile iamlost*
restore> extract
You have not read any volumes yet.
Unless you know which volume your files are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
```

卷（磁带）是从1而不是从0开始记数的，所以，对于在单独一卷磁带上进行的转储而言，可以指定1。当 **restore** 命令询问是否需要为“.”设置属主和模式时，它是在询问是否应该设置当前目录去匹配磁带的根。除非您是在恢复整个文件系统，否则可能不需要这么做。

restore 完成之后，就必须把该文件交给用户 janet：

```
# cd /var/restore
# ls janet
iamlost
# ls ~janet
afile bfile cfile
# cp -p janet/iamlost ~janet/iamlost.restored
# chown janet ~janet/iamlost.restored
# rm -rf /var/restore
# mail janet
Your file iamlost has been restored as requested and has
been placed in /users/janet/iamlost.restored.

Your name, Humble System Administrator
```

有些管理员喜欢把文件恢复到特定的目录中，让用户手工拷走他们自己的文件。在这种方案中，管理员必须通过检验这些文件的所有权和权限来保护所恢复文件的机密性。如果选择使用这样一个系统，那么请记住要经常清理该目录。

如果使用 **rdump** 命令创建了一个备份，但不能使用 **restore** 命令从该备份中恢复文件，那么请试着运行一下 **rrestore** 命令。为了将出问题的几率减到最小，应该使用写磁带时使用的那台主机去读磁带。

通常，**restore -i** 是从一次转储备份中恢复少许文件或者目录最简单的方法。但是，如果磁带设备一次不能后移一个记录（某些8毫米磁带机有的一个问题），那么它就不能工作。如果 **restore -i** 命令失败了，那么在退出窗口之前试一下 **restore -x** 命令。**restore -x** 需要您在命令行中指出所要恢复文件的完整路径（相对于转储根目录的路径）。下面的几条命令使用 **restore -x** 来重复前面的例子：

```
# mkdir /var/restore
# cd /var/restore
# ssh tapehost mt -f /dev/nst0 fsf 3
# rrestore -x -f tapehost:/dev/nst0 ./janet/iamlost
```

9.4.2 恢复整个文件系统

幸运的话，在一次系统故障之后您不必恢复整个文件系统。但是，这种情况偶尔也会出现。在试图恢复文件系统之前，必须首先确保造成系统破坏的任何问题都已经得到了解决。如果花费无数小时

[8] **iamlost** 旁边的星号表明已经标记它要提取出来。

www.TopSage.com

去转那些磁带，结果却只是再次丢失文件系统，这样就没意义了。

在开始一次完整的恢复之前，必须创建和安装目标文件系统。请参看第 7 章中的内容，了解关于如何准备文件系统的更多信息。为了开始进行恢复，应该使用 **cd** 命令进入到新文件系统的安装点，然后将最近一次 0 级转储备份的第一卷磁带放入到磁带机中，并且键入 **restore -r**。

restore 命令将提示安装该转储备份中的每卷磁带。在恢复了 0 级转储备份之后，就应该安装和恢复增量转储备份了。应该按照创建它们的顺序来恢复增量转储备份。因为各个转储备份间有冗余，所以可能没有必要恢复每一个增量转储备份。下面是决定恢复哪些转储备份的算法。

步骤 1：恢复最近一次 0 级转储备份。

步骤 2：恢复在刚才所恢复的转储备份之后创建的所有转储备份中级别最低的一次转储备份。如果在那个级别中有多次转储备份，那么应该恢复最近的那一次备份。

步骤 3：如果这就是上一次转储备份，那么就完成了任务。

步骤 4：否则，返回到步骤 2 中。

这里有一些转储顺序的例子。您只需要恢复以黑体字显示的级别即可。

0 0 0 0 0 0
0 5 5 5 5
0 3 2 5 4 5
0 9 9 5 9 9 3 9 9 5 9 9
0 3 5 9 3 5 9

让我们来看一个完整的命令序列。如果最近的转储备份是在前面讲到的“适中”安排中每年 0 级备份之后第一个月备份的，那么用于恢复位于物理设备**/dev/sdal** 上的**/home** 所需使用的命令看起来和下面的类似（设备名随硬件的不同而不同）：

```
# /etc/mke2fs -j /dev/sda1 QUANTUM_PD1050S
# /etc/mount /dev/sda1 /home
# cd /home
/* Mount first tape of level 0 dump of /home. */
# restore -r
/* Mount the tapes requested by restore. */
/* Mount first tape of level 3 monthly dump. */
# restore -r
```

请参看第 7 章中的内容，了解关于 **mke2fs** 和 **mount** 命令的更多信息。

如果在一卷转储磁带上有多个文件系统，那么就不得不在每次运行 **restore** 命令之前使用 **mt** 命令来向前跳到正确的文件系统处。请参看 9.7 节中的内容，了解对 **mt** 的说明。

这几条命令将把文件系统恢复到做 5 级转储备份时的状态，而且所有被删除的文件都会被恢复。当您正在恢复一个活动的文件系统，或者正在恢复的信息几乎已填满了一块硬盘的时候，这个问题就会变得非常讨厌了。因为文件系统已经被恢复出来的文件填满了，所以 **restore** 命令很可能会失败。

9.5 为系统升级而执行转储和恢复

我们建议在对主操作系统进行升级的时候，先使用 0 级转储来备份所有的文件系统，如果可能还要使用 **0** 级转储备份恢复它们。只有当新的操作系统使用了不同的文件系统格式或者您改变了硬盘的分区时，才需要恢复。但是您必须进行备份，以防在升级过程中可能发生的任何问题。如果新版本的操作系统不能让您满意的话，一套完整的备份还将给您重新安装老的操作系统的机会。幸好现如今大多数发行版本都采用了渐进的方式升级系统，所以您不大可能需要这些磁带。

www.TopSage.com

一定要确保备份和恢复在根（/）目录或者/usr 目录中任何系统特定的文件，比如/etc/passwd、/etc/shadow 或者/usr/local。Linux 系统中混乱的目录组织结构把本地文件和厂商发布的文件混合在了一起，这使得挑选出自己本地计算机专门定制信息的操作变得非常困难。

应该在升级之后也立刻进行一整套 0 级转储操作。大多数厂商的升级过程都会把系统文件的修改日期设置为它们被复制到母盘介质上的时间，而不是当前时间。因此，在出现一次系统崩溃时，相对于升级之前 0 级备份所做的增量转储还不足以将系统恢复到其升级之后的状态。

9.6 使用其他存档程序

dump 命令不是用于把文件存档到磁带中的唯一程序。但在通常情况下，它是备份整个系统的最有效方式。**tar**、**cpio** 和 **dd** 命令也能够把文件从一个介质转移到另外一个介质中。

9.6.1 tar: 给文件打包

tar 命令读取多个文件或者目录，并把它们打包成一个文件，通常情况下是一个磁带文件。**tar** 是备份任何预计近期要恢复的文件的一种有用的方法。例如，如果某个用户将要离开 6 个月，而系统缺少硬盘空间，那么就可以使用 **tar** 命令把用户的文件存放在磁带上，然后把它们从硬盘上删除。

对于把目录树从一个位置移动到另外一个位置这样的操作，尤其是当您以超级用户身份进行复制（**tar** 命令将保存文件的所有权信息）的时候，**tar** 命令也是非常有用的。例如，命令：

```
tar -cf - fromdir | ( cd todir ; tar --atime-preserve -xpf - )
```

将在 *todir* 中创建目录树 *fromdir* 的一个副本。应该避免在 *todir* 参数中使用“..”，因为符号链接和自动安装工具可能会让它的含义与您所期望的不同。我们已经吃过几次这样的亏了。

默认情况下，**tar** 命令不会对符号链接所指向的真正文件做备份，但可以指示它去这么做。也可以指定 **tar** 只包含自给定日期以来修改过的文件，这对于创建您自己的增量备份方案来说很有帮助。请参考 **tar** 手册页，了解这项功能以及别的好功能。

在很多非 Linux 版本的 **tar** 命令中有一个问题，这就是路径名被限制在 100 个字符之内。这一限制可以妨碍 **tar** 命令对很深的目录层次进行存档。如果您在 Linux 系统上创建了 **tar** 的存档文件，并把它转到了别的系统上，那么要记住使用标准 **tar** 命令的用户有可能无法读取您所创建的磁带或者文件。

tar 命令中的**-b** 选项可以让您在写磁带时指定一个“分块因子”以供使用。分块因子是以 512 字节的块为单位来规定的。它决定了在执行写入操作之前 **tar** 命令会在内部缓冲多少数据。除非分块因子被设置为一个特定值，否则有些 DAT 设备不能正常工作，但其他设备不需要这项设置。

在有些系统中，特定的分块因子可以获得比其他因子更好的性能。这种最优分块因子的取值范围很广，具体取决于计算机和磁带设备。在很多情况下，您都不会感觉到任何速度上的差别。当您有疑问的时候，可以试用一个值为 20 的分块因子。

除非您在创建原始存档文件时使用**-S** 选项，否则 Linux 的 **tar** 命令会扩大文件中的孔洞。Linux 的 **tar** 不大能容忍磁带出错。

9.6.2 cpio:古老的存档工具

cpio 在功能上和 **tar** 相似。它可以追溯到 UNIX 的最初时期，但现在很少使用了。和 **tar** 一样，**cpio** 命令能够用于移动目录树。如下命令：

www.TopSage.com

```
find fromdir -depth -print | cpio -pdm todir
```

将在 *todir* 中建立目录树 *fromdir* 的一个副本。GNU 版本的 **cpio** 命令允许使用多卷磁带，但是大多数的 **cpio** 版本不行。只有超级用户才能够复制特定的文件。即便您非常熟悉另一个系统上的 **cpio**，我们还是要建议您仔细阅读手册页，因为在各个系统之间，其选项有很大的不同。

9.6.3 dd：处理位流

dd 是文件复制和转换程序。除非告诉 **dd** 命令去进行某种转换操作，否则它只进行从输入文件到输出文件的复制工作。如果用户给您一卷在某些非 Linux 系统上写入的磁带，那么 **dd** 命令可能是读取它的唯一方法。

dd 命令在历史上的一种应用就是创建一个完整文件系统的副本。但现在更好的选择是使用 **mke2fs** 命令创建目标文件系统，然后运行 **dump** 命令导出给 **restore** 命令。如果使用不正确的话，**dd** 命令有时候能够破坏分区信息。它只能在大小完全相同的两个分区之间复制文件系统。

dd 命令还能够用于磁带复制。有两个磁带驱动器，比如说/**dev/st0** 和/**dev/st1** 的话，您可以使用命令：

```
$ dd if=/dev/st0 of=/dev/st1 cbs=16b
```

只有一个驱动器（/**dev/st0**）的话，可以使用下面的一系列命令：

```
$ dd if=/dev/st0 of=tfile cbs=16b
/* 更换磁带. */
$ dd if=tfile of=/dev/st0 cbs=16b
$ rm tfile
```

当然，如果您只有一个磁带驱动器，就必须要有足够的硬盘空间来存储整卷磁带的一个映像。

9.7 在一卷磁带上使用多个文件

在实际使用中，一卷磁带包含了一长串数据。但是，在一卷磁带上存储不止一件“东西”往往是有用的，所以磁带机和它们的 Linux 驱动程序共同提供了一个更好的结构。当 **dump** 命令或者某些其他命令把一个字节流写到一卷磁带设备上，然后关闭设备文件的时候，会自动地在磁带上放置一个“EOF（文件末尾）”标记。这个标记把该字节流和随后写入的其他字节流区分开来。当重新读入该字节流时，读取操作会自动停在 EOF 标记处。

您可以使用 **mt** 命令来把磁带定位在特定的流，或者“文件集”（照 **mt** 的叫法）处。如果将多个文件（比如说，多个转储文件）存在一卷磁带上，则 **mt** 命令尤其有用。它的出错信息，也是 Linux 工具中最有趣的。这个命令的基本格式是：

```
mt [-f tapename] command [count]
```

command 有很多不同的选择。它们随平台的不同而不同，所以我们只讨论进行备份和恢复所必需的几种。

rew 将磁带倒带到开始位置。

offl 将磁带脱机。在大多数磁带机上，这个命令将导致磁带倒带并从驱动器中弹出。大多数转储脚本都会在完成转储操作之后使用这条命令来弹出磁带，这样可以给出每项工作都已经正确完成的明确指示。

www.TopSage.com

status 输出磁带驱动器的当前状态信息（是否装入了磁带等）。

fsf [*count*] 磁带快速前进。如果没有给出 *count*，那么 **fsf** 将向前跳过一个文件。用一个数字做参数的话，它就会跳过指定数目的文件。使用这个命令在包含有多次转储的一卷磁带上向前跳到正确的文件系统。

bsf [*count*] 应该倒后 *count* 个文件。这条指令的准确结果取决于磁带机硬件和与之相关联的驱动程序。在有些情况下，当前的文件被计算在内。而在别的情况下，当前文件不包含在内。在有些设备上，**bsf** 不进行任何操作（保持沉默）。如果您前进到了磁带上过远的位置，那么最好的方法就是执行 **rew**，然后从头再次开始。

请参考您的手册页以了解 **mt** 所支持命令的确切清单。

如果您运气够好，有一个自动控制的磁带库，那么就能通过安装 **mtx** 软件包（**mt** 命令的一个增强版本）来控制它的磁带更换设备。例如，我们用 **mtx** 控制我们特棒的 HP 6x24 DAT 磁带机系统实现无人更换磁带。瞧，根本不用动手！

9.8 Bacula

Bacula 是一种企业级的客户机/服务器模式的备份解决方案，它能够通过网络来管理文件的备份、恢复和核实工作。Bacula 在各种 UNIX 和 Linux 系统上运行，其中也包括我们所举例的几种 Linux 发行版本在内。Bacula 还能备份来自多种操作系统（包括微软的 Windows）的数据。

在本书的上一版里，Amanda 是我们最喜爱的非商业备份工具。如果您需要了解 Amanda 的资料，可以查阅本书的第一版，或者访问 www.amanda.org。下面的特色说明了 Bacula 为什么成为我们最喜爱的备份工具。

- 模块化设计；
- 能够备份 UNIX、Linux 和 Windows 系统；
- 支持 MySQL、PostgreSQL 或者 SQLite 作为后端数据库；
- 支持易用的菜单驱动的命令行控制台；
- 遵守开放源代码许可证；
- 备份可以横跨多个磁带卷；
- 服务器可以在多种平台上运行；
- 可以给备份的每个文件都创建 SHA1 或者 MD5 的签名文件；
- 校验备份工作；
- 支持磁带库和自动换带机；
- 在备份工作前后可以执行脚本或者命令；
- 对整个网络集中备份管理。

9.8.1 Bacula 的模型

为了部署 Bacula，应该了解它的主要组件。图 9.1 画出了 Bacula 的基本体系结构。

Bacula 的控制器是协调备份、恢复和校验操作的守护进程。用户可以使用 Bacula 的控制台向控制守护进程提交备份或者恢复任务。用户还可以要求控制守护进程查询 Bacula 的存储守护进程或者位于客户计算机上的文件服务守护进程。

用户通过 Bacula 控制台与控制守护进程打交道，Bacula 控制台既能以 GNOME GUI 的方式运行，也可以按命令行的方式执行。控制台能够在任何地方运行，它不必和控制守护进程放在同一台的计算

www.TopSage.com

机上。

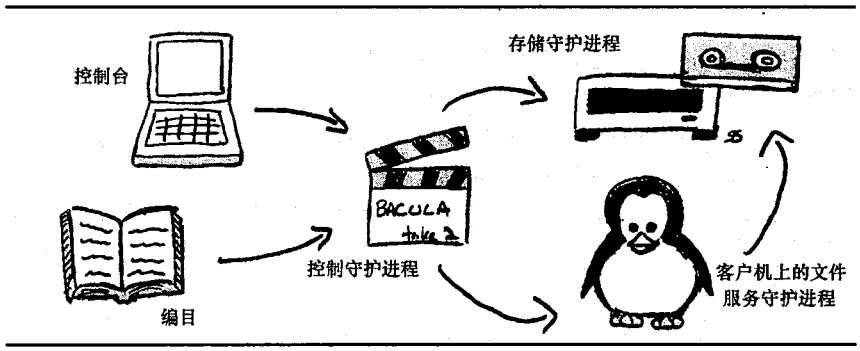


图 9.1 Bacula 的组件以及它们之间的关系

存储守护进程是 Bacula 读写磁带或者其他备份介质的组件。这一服务必须运行在连接有供备份使用的磁带机或者存储设备的计算机上，但是不必把它和控制守护进程安装到同一台机器上（虽然可以安装在同一台机器）。

Bacula 的文件守护进程在每个要备份的系统上运行。在执行备份任务的时候，每种 Bacula 所支持的操作系统上的文件守护进程实现都把适当的文件数据和属性发送给存储守护进程。

最后一个 Bacula 组件是编目，Bacula 把所备份的每个文件和卷的信息都保存在这个关系数据库里。Bacula 的编目让恢复过程快速而高效，因为完整的备份历史信息都能在线访问到。Bacula 在开始读取一盘磁带之前，就知道恢复某一组特定的文件需要哪些存储卷。Bacula 现在支持 3 种不同的数据库：MySQL、PostgreSQL 和 SQLite。编目数据库无需和控制守护进程在同一台服务器上。

还有一种可选的组件叫做 Bacula Rescue CD-ROM（Bacula 救急光盘）。这个组件是一个可以单独下载的软件包，它创建一张可启动的救急光盘，供 Linux 系统在进行灾难恢复期间使用。光盘包含系统上文件守护进程的一个静态链接的版本，还有定制过的 shell 脚本，脚本有系统的磁盘、内核以及网络接口的配置信息。如果 Linux 系统发生灾难性故障，可以使用 Bacula 的救急光盘来启动系统，重新对硬盘分区，并且连接到 Bacula 的控制守护进程，通过网络做一次完整的系统恢复。

9.8.2 设置 Bacula

因为 Bacula 的结构复杂，具有一组高级特性，而且采用模块化设计，所以能通过多种方式建立全站点的备份方案。在这一节，我们看一下 Bacula 的基本配置。

一般来说，让 Bacula 启动并运行起来需要 5 个步骤：

- 安装 Bacula 所支持的一种第三方数据库以及 Bacula 的守护进程；
- 配置 Bacula 的守护进程；
- 安装和配置客户机上的文件守护进程；
- 启动 Bacula 的守护进程；
- 用 Bacula 控制台把介质加入存储池。

我们给出的例子包括两台计算机 harp 和 bull。计算机 harp 是客户机，它只运行了一个文件守护进程。其余的 4 个 Bacula 组件（控制器守护进程、存储守护进程、编目数据库和控制台）都运行在服务器 bull 上。存储设备是一台 SureStore LTO1 磁带库。我们的备份部署包括每周的完整备份和每天的增量备份。编目采用 MySQL 数据库。

www.TopSage.com

9.8.3 安装数据库和 Bacula 的守护进程

在我们开始安装 Bacula 之前，必须先安装用于 Bacula 编目的后端数据库。为了最方便地与 Bacula 集成，我们推荐把 MySQL 安装在它的默认配置路径里。

在建立一个备份平台的时候，必须考虑稳定性和可靠性，所以一旦安装好了数据库，我们建议从 Bacula 的网站下载和安装最新的稳定的源代码版本。源代码中的 **docs** 目录里有一步一步安装 Bacula 的文档。这份文档也可以从 www.bacula.org 在线访问，在那里有 HTML 和 PDF 两种格式的文档，以及很有帮助的教程和开发人员指南。

在解开源代码包之后，我们运行：

```
./configure --with-mysql --prefix=/etc/bacula
```

之后执行 **make** 来编译出二进制结果，最后用 **make install** 完成安装。

安装好 Bacula 之后，下一步要创建实际用的 MySQL 数据库，并在数据库内建表。我们先要确认 MySQL 正在运行，然后使用 **cd** 命令进入到安装目录（**/etc/bacula**）里，运行 **make install** 执行时所创建的 3 个 shell 脚本。脚本 **grant_mysql_privileges** 给 Bacula 用户分配 MySQL 访问权限。脚本 **create_mysql_database** 创建 Bacula 的数据库，脚本 **make_mysql_tables** 在数据库内创建所需的表。

9.8.4 配置 Bacula 的守护进程

在开始更详细地讨论我们所举的这个 Bacula 配置例子之前，我们先定义一些 Bacula 的关键术语。

- “作业（job）”是 Bacula 运行活动的单位。它们分为两类：备份和恢复。
- “池（pool）”是保存作业的物理介质组。我们所举的这个配置例子用到了两个池，一个用于完整备份，另一个用于增量备份。
- “文件集（fileset）”是分区和文件的列表。在备份或者恢复作业中既能包括若干文件集也能排除若干文件集。
- “引导（bootstrap）”文件是 Bacula 创建的一个特殊的文本文件，它包含有应该备份的文件的信息。在执行恢复命令期间，或者在执行备份命令期间给备份作业用了 Write Bootstrap 参数的时候，都会创建引导文件。
- “消息（message）”是守护进程之间涉及守护进程和作业状态的通信信息（实际上就是日志项）。消息也可以通过电子邮件发送，并写入日志文件。

在安装期间，配置文件位于由**--prefix** 选项指定的目录下，一般这个目录是**/etc/bacula**。在下一节，我们将介绍控制守护进程（**bacula-dir.conf**）、存储守护进程（**bacula-sd.conf**）和控制台（**bconsole.conf**）的配置选项。

在我们所举的例子里，我们没有覆盖所有可能的配置参数。我们代之以在每一小节的开头先做概述，然后再指出一些我们认为特别有用或者不好掌握的参数。

9.8.5 bacular-dir.conf：配置控制文件

文件 bacula-dir.conf 是 Bacula 的配置文件中最复杂的一个。下面我们看看它的 10 个逻辑段：`Director`、`Catalog`、`JobDefs`、`Job`、`FileSet`、`Schedule`、`Pool`、`Client`、`Storage` 和 `Messages`。每个配置段及其参数的详细解释都能在在线文档中找到。

每个资源段都用大括号括起来。在所有的 Bacula 配置文件中，注释都以#开头。

www.TopSage.com

```
# Sample Bacula director configuration file, /etc/bacula-dir.conf

Director {
    Name = bull-dir
    DIRport = 9101
    Query File = "/etc/bacula/query.sql"
    Working Directory = "/var/Bacula/working"
    Pid Directory = "/var/run"
    Maximum Concurrent Jobs = 1
    Password = "B@cu1@Lik3s,fRu17"
    Messages = Standard
}
```

Director 资源多少有些像 Bacula 大海上的母舰。它的参数规定了控制器的名字和基本行为。它的选项设置了其他守护进程与控制器进行通信所采用的通信端口、控制器保存其临时文件的位置，以及控制器一次能处理的并发作业数量。

参数 Password 规定了控制台程序用来向控制器证实身份的口令。在控制台配置文件（**bconsole.conf**）的 Director 资源段里必须设置一样的口令。虽然这个口令在配置文件里显示为明文，但是这个口令绝不会通过网络传输。

在我们所举的配置例子中，控制器和控制台都在同一台机器上。不过，两个配置文件里仍然都要有口令。

```
# Generic catalog service

Catalog {
    Name = ATE
    DBName = bacula;
    user = bacula;
    password = "fRu17,B0wL"
}
```

Catalog 资源段规定了 Bacula 保存其操作记录的数据库。为了简单起见，我们给所有的作业定义了一个编目。但是 Bacula 的确支持多个编目，如果要为每个客户组维护一个独立的数据库，这个功能就能派上用处。目前，所有的编目数据库都必须在物理上的同一个数据库服务器中。

参数 DBName 是编目数据库的名字。这个数据库在安装期间按照传递给 **configure** 的数据库参数来设置。在 Bacula 源代码中的 **src/cats** 目录下，可以找到 Bacula 预先写好的创建数据库的脚本。

```
# Job definitions

Job {
    Name = "harp"
    Level = Full
    Write Bootstrap = "/atrust/admin/backups/bootstraps/harp.bsr"
    Client = harp
    File Set = harp
    Storage = SureStore
    Pool = SSFullPool
    Incremental Backup Pool = SSIncrementalPool
    Schedule = Nightly
}
```

Job 资源定义了一个特定备份作业默认的参数。一般来说，每个客户有一个 Job 定义。大多数参数都能顾名思义，但是有几个特性要多讨论一下。

Write Bootstrap 参数告诉 Bacula 把有关备份的引导信息写在什么地方，供恢复作业期间使用。引导文件列出了一次恢复作业所需的文件和卷。它们不是一定要有，但是我们强烈建议要有。在完整备份期间会覆盖引导文件，而在增量备份期间则会在引导文件后面追加信息。

www.TopSage.com

参数 Client、File Set、Storage、Pool 和 Schedule 都是对 **bacula-dir.conf** 文件后面所定义的资源在前面的引用。

```
# File set definitions

FileSet {
    Name = "harp"
    Include {
        Options {
            signature=MD5
            compression=GZIP
        }
        File = "/"
        File = "/boot"
        File = "/usr"
        File = "/usr/local"
        File = "/var"
    }
    Exclude = { /proc /tmp /.journal /.fsck }
}
```

FileSet 资源定义了一个备份作业所包含的或者所排除的文件和目录。每个文件集都定义了多个 Include 和 Exclue 参数以及各自的 Options。在默认情况下，Bacula 递归地备份目录，但是不会跨越分区。注意在单独的 File 参数里列出想要备份的所有分区。

这里支持多个 Options，包括正则表达式和通配符。值得一提的有两个选项，一个是 signature（设为 SHA1 或者 MD5），它计算所备份的每个文件的散列值，还有一个是 compression，设置在把数据写入磁带之前先进行压缩处理。signature 选项会增加备份的 CPU 开销，但是在怀疑有安全问题的情况下则有使用价值。

```
Schedule {
    Name = "Nightly"
    Run = Level=Full Pool=SSFullPool 1st-5th tue at 20:10
    Run = Level=Incremental Pool=SSIncrementalPool wed-mon at 20:10
}
```

Schedule 资源规定了备份作业的时间表。自动备份作业需要参数 Name 和至少一个 Run 参数，但是在一个 Schedule 资源里可以包含多个 Run 参数。在本例中，在每月的第 1 和第 5 个星期二晚上 8:10 做完整备份，在每周三到周一的晚上 8:10 做增量备份。

Run 参数只需要日期和时间说明，但是由本例可见，可以加入更多的参数值。这些值会覆盖在 Job 说明中设置的默认参数。我们在这里设置了每次备份使用的备份级别（Level）和存储池（Pool）。

参考 Bacula 的在线文档，了解日期/时间说明关键字和作业参数的完整清单。

```
# Pool definitions -- first pool is default

Pool {
    Name = SSFullPool
    Pool Type = Backup
    Recycle = yes
    Accept Any Volume = yes
}

Pool {
    Name = SSIncrementalPool
    Pool Type = Backup
    Recycle = yes
    Accept Any Volume = yes
}
```

www.TopSage.com

Pool 资源把备份介质（一般为磁带）划分为给特定备份作业所使用的组。在本例中，我们每周做一次完整备份，所以我们创建了两个 Pool 资源。许多选项能调节 Pool 资源的运行，其中特别有用的两个参数是 Recyle 和 Accept Any Volume。如果将 Recyle 参数设为 yes，那么 Bacula 在它需要给一个备份作业追加卷的时候会自动使用已被清干净的卷（设置了循环使用[recyle]标志）。Accept Any Volume 参数指出 Bacula 是否能把数据写入存储池里任何一个可以追加的卷。如果这个选项设为 no，那么 Bacula 按次序填满卷，不会不按次序接受一个可以追加的卷。

```
Client {
    Name = harp
    Address = 192.168.7.2
    FDPort = 9102
    Catalog = ATE
    Password = "Ch@ch1s@Fru17"
    File Retention = 3 months
    Job Retention = 36 months
    AutoPrune = yes
}
```

Client 资源标识要备份的计算机。每台计算机要一个 Client 资源。File Retention 和 Job Retention 参数规定了在编目中保留这个客户机多长的文件和作业记录。如果设置了 AutoPrune 参数，那么过期的数据就会从编目中删除。削减工作只会影响到编目里的记录，不会影响备份磁带上保存的实际文件。

```
# Definition of file storage devices

Storage {
    Name = SureStore
    Address = bull.atrust.com
    SDPort = 9103
    Password = "Fru1t&V3gg1es"
    Device = SureStoreDevice
    Autochanger = yes
    Media Type = LTO1
}
```

Storage 资源说明了怎样同存储守护进程进行通信，该进程控制着备份设备。在我们所举的例子里，我们把一台 SureStore LTO1 磁带库接到了我们的主 Bacula 服务器上。存储守护进程有它自己的配置文件，其内容如下：

```
Messages {
    Name = Standard
    mailcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
        bacula@atrust.com\" -s \"Bacula: %t %e of %c %l\" %r"
    operatorcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
        bacula@atrust.com\" -s \"Bacula: Intervention needed for %j\" %r"
    mail = Bacula@atrust.com = all, !skipped
    operator = bacula-pager@atrust.com = mount
    console = all, !skipped, !saved
    append = "/var/log/bacula.log" = all, !skipped
}
```

Messages 资源告诉 Bacula 如何处理 Bacula 守护进程产生的特定消息类型。可以定义多个 Messages 资源，然后分配给 Job 资源中的特定作业。

这个资源类型可配置性很强，从在线文档中可以找到变量和命令的完整清单。上述配置将有关守护进程和作业的状态信息发给控制台和一个标志日志文件，同时还会通过电子邮件散发。

www.TopSage.com

9.8.6 bacula-sd.conf：配置存储守护进程

存储守护进程接受从文件守护进程来的数据，把数据传给实际的存储介质（在恢复过程中，数据的流向正好相反）。在 **bacula-sd.conf** 这个文件中必须定义 4 种资源：`Storage`、`Device`、`Messages` 和 `Director`。下面是一个完整的配置样例：

```
# Storage daemon configuration file, bacula-sd.conf

Storage {
    Name = bull-sd
    SDPort = 9103
    Working Directory = "/var/bacula/working"
    Pid Directory = "/var/run"
}

Device {
    Name = SureStoreDevice
    Media Type = LTO1
    Archive Device = /dev/nst0
    Autochanger = yes
    Changer Device = /dev/sg0
    Changer Command =  "/etc/bacula/mtx-changer %c %o %S %a %d"
    AutomaticMount = yes;
    Always Open = yes;
    Removable Media = yes;
    Random Access = no;
}

Messages {
    Name = Standard
    director = bull-dir = all
}

Director {
    Name = bull-dir
    Password = "Fru1t&V3gg1es"
}
```

存储守护进程的配置文件必须只包含一个 `Storage` 资源。这些资源都很直观。它们定义了基本的工作参数，比如守护进程的网络端口和工作目录，还标识出存储守护进程应该把消息发送给哪个控制器守护进程。

`Device` 资源刻画了实际的备份设备。在我们所举的例子里，这个备份设备是一台有自动换带装置的 LTO（Linear Tape-Open）驱动器。您可以定义多个 `Device` 资源。`Achive Device` 参数给磁带机取了设备名。注意**/dev/nst0** 是一种非自动倒带设备，它基本上就是您所想要的设备。自动磁带更换机有它自己的设备文件，除了指定那个文件之外，我们还要设置 `Autochanger` 参数。参数 `Always Open` 告诉 **Bacula**，除非系统管理员特意要求卸载设备，否则就保持设备一直打开的状态。这个选项节省了时间和磁带的磨损，因为它避免了在两次作业之间执行倒带和定位的命令。

9.8.7 bconsole.conf：配置控制台

用户使用控制台程序和控制器进行通信来安排作业、检查作业状态或者恢复数据。在安装目录下输入**./bconsole** 就可以启动控制台。

www.TopSage.com

文件 **bconsole.conf** 告诉控制台程序如何与 Bacula 的控制器守护进程通信。这个文件中的参数必须和控制器的配置文件（**bacula-dir.conf**）里的 Director 资源相对应，而 address 参数则例外。

```
# Console configuration file, bconsole.conf

Director {
    Name = bull-dir
    DIRport = 9101
    address = bull.atrust.com
    Password = "B@cu1@Lik3s,fRu17"
}
```

9.8.8 安装和配置客户端的文件守护进程

在执行备份和恢复作业期间，备份客户机的文件守护进程会同 Bacula 存储守护进程进行通信。在每台要用 Bacula 备份的计算机上都必须安装和配置这个守护进程。

对于 Windows 客户机来说，可以从 Bacula 的网站下载预先编译后的二进制程序。Bacula 能很好地备份 Windows 的数据文件，但是却不擅长做一个 Windows 系统的 0 级备份。遗憾的是，Bacula 对于 Windows 打开文件的锁、注册表和系统状态都没有概念，所以必须采取更多的步骤才能保证在执行 Bacula 的转储之前保存好了这些数据。在线文档对解决这些问题提供了很好的帮助。

在 UNIX 和 Linux 系统上，可以通过把 Bacula 的源代码树拷贝到每台客户机，并且运行下面的命令来安装这个守护进程：

```
./configure --enable-client-only --prefix=/etc/bacula
```

接着再运行 **make** 和 **make install**。

安装好二进制程序之后，编辑**/etc/bacula/bacula-fd.conf** 文件配置文件守护进程：

```
# File daemon configuration for harp, bacula-fd.conf

Director {
    Name = bull-dir
    Password = "Ch@ch1s@Fru17"
}

# "Global" file daemon configuration specifications

FileDaemon {
    Name = harp
    FDport = 9102
    Working Directory = /var/bacula/working
    Pid Directory = /var/run
}

# Send all messages except skipped files back to the director

Messages {
    Name = Standard
    director = bull-dir = all, !skipped
}
```

这个配置文件分成 3 个部分。第一部分由 `Director` 资源构成，它告诉文件守护进程哪个控制器能够从这个客户机的文件守护进程来布置执行备份作业。`Director` 资源还包含 `Password` 参数，它必须和控制器自己的配置文件中 `Client` 资源里列出的口令一样。第二部分是 `FileDaemon` 资源，它给客户机起名字，并且指定文件守护进程在哪个端口上监听从控制器守护进程发来的命令。最后一

www.TopSage.com

部分是 Messages 资源，它规定了本地的消息如何处理。

9.8.9 启动 Bacula 的守护进程

现在我们已经安装好了我们的服务器守护进程，也配置好了客户机，接下来的一步就是运行服务器安装目录下的启动脚本（**./bacula start**）让守护进程开始运行了。在每个客户机上也能用同一命令来启动文件守护进程。这个脚本应该也链接到了 **rc** 和 **init.d** 目录下适当的运行级启动脚本上。

Bacula 守护进程开始运行之后，您就可以使用控制台程序（在安装目录下的 **bconsole**）检查它们的状态、向存储池添加介质、以及执行备份和恢复作业。任何一台计算机，只要它已经正确地安装和配置好了，就可以在它上面运行 **bconsole**。

```
$ sudo ./bconsole
Password: <entered password>
Connecting to Director bull:9101
1000 OK: bull-dir Version: 1.38.11 (29 June 2006)
Enter a period to cancel a command.
```

使用控制台的 **help** 命令就能看到它所支持的命令的一份完整清单。

9.8.10 向存储池添加介质

在可以运行备份作业之前，需要给一些磁带做好标签，再把它们分配给控制器的配置文件中所规定的存储池。完成这项工作需使用控制台的 label 命令。

```
* label
Automatically selected Storage: SureStore
Enter new Volume name: 003061L1
Enter slot (0 for none): 14
Defined Pools:
     1: SSFullPool
     2: SSOneOffPool
...
Select the Pool (1-5): 1
Connecting to Storage daemon SureStore at bull.atrust.com:9103
...
Sending label command for Volume "003061L1" Slot 14 ...
...
3001 Mounted Volume: 003061L1
3001 Device /dev/nst0 is mounted with Volume "003061L1"
```

在这个例子中，自动磁带更换机的 14 槽被称为 003061L1，它被分配给了存储池 SSFullPool。使用 **list media** 命令来核实磁带已经添加到了正确的存储池中，并标记为可追加磁带。

9.8.11 执行一次手工备份

使用控制台的 **run** 命令就能执行一次手工备份。这条命令不需要参数。控制台显示在控制器的配置文件里定义的所有备份作业。您可以根据控制台以菜单方式提供的提示，修改在 run 命令中的任何选项。

下面这个例子演示了一次对服务器 harp 的手工完整备份作业，备份采用了我们的配置文件中的默认设置：

www.TopSage.com

```
$ sudo ./bconsole
...
Connecting to Director bull:9101
1000 OK: bull-dir Version: 1.34.6 (28 July 2004)
Enter a period to cancel a command.
* run
A job name must be specified.
The defined Job resources are:
     1: harp
     2: RestoreFiles
Select Job resource (1-2): 1
Run Backup job
JobName:  harp
FileSet:  harp
Level:  Full
Client:  harp
Storage:  SureStore
Pool:  SSFullPool
When:  2006-07-08 13:14:24
Priority:  10
OK to run? (yes/mod/no): yes
Run command submitted.
```

成功向控制器提交备份作业之后，就可以用控制台的 **status** 命令跟踪它的状态，还可以使用 **messages** 命令随消息的到来获得详尽的更新信息。根据设置系统的 Messages 资源方式不同，可以向 Bacula 的管理员发送一份详细的汇总报告。在备份作业成功完成之后，**messages** 命令产生的输出如下，它包括许多有用的汇总信息。

```
* messages
08-Jul-2006 14:21 bull-dir: Start Backup JobId 5216, Job=harp.2006-07-08_14.21.03
08-Jul-2006 14:23 bull-sd: 3301 Issuing autochanger "loaded drive 0" command.
08-Jul-2006 14:23 bull-sd: 3302 Autochanger "loaded drive 0", result is Slot 6.
08-Jul-2006 14:23 bull-sd: 3303 Issuing autochanger "unload slot 6, drive 0" command.
08-Jul-2006 14:24 bull-sd: 3304 Issuing autochanger "load slot 14, drive 0" command.
08-Jul-2006 14:25 bull-sd: 3305 Autochanger "load slot 14, drive 0", status is OK.
08-Jul-2006 14:25 bull-sd: Volume "003048L1" previously written, moving to end of data.
08-Jul-2006 14:25 bull-sd: Ready to append to end of Volume "003048L1" at file=7.
08-Jul-2006 14:54 bull-dir: Bacula 1.34.6 (28Jul04): 08-Jul-2006 14:54
JobId:                  5216
Job:                    harp.2006-07-08_14.21.03
Backup Level:           Full
Client:                 harp
FileSet:                "harp" 2006-01-06 20:37:06
Start time:             08-Jul-2006 14:21
End time:               08-Jul-2006 14:54
FD Files Written:       176,451
SD Files Written:       176,451
FD Bytes Written:       12,968,821,897
SD Bytes Written:       12,993,427,904
Rate:                   6379.2 KB/sSoftware Compression:None
Volume name(s):         003048L1
Volume Session Id:      263
Volume Session Time:    1149888981
Last Volume Bytes:      13,298,706,752
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:  OK
```

www.TopSage.com

大家网 TopSage.com

```
SD termination status:    OK
Termination:              Backup OK
08-Jul-2006 14:54 bull-dir: Begin pruning Jobs.
08-Jul-2006 14:54 bull-dir: No Jobs found to prune.
08-Jul-2006 14:54 bull-dir: Begin pruning Files.
08-Jul-2006 14:55 bull-dir: Pruned 2,206 Files from 48 Jobs for client harp from catalog.
08-Jul-2006 14:55 bull-dir: End auto prune.
```

9.8.12 执行一次恢复工作

要恢复文件，可以启动控制台，然后运行 **restore** 命令。**restore** 命令和 **run** 命令一样，也采用了菜单方式。它首先帮助用户确定恢复目标文件所需要读取的作业。**restore** 命令给用户提供了几种方法来指定相关作业 ID。选择好一组作业之后，接下来就可以从那些作业中选择要恢复的文件。

```
* restore
To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of comma separated JobIds to select
     4: Enter SQL list command
     5: Select the most recent backup for a client
     6: Select backup for a client before a specified time
     7: Enter a list of files to restore
     8: Enter a list of files to restore before a specified time
     9: Cancel
Select item:  (1-9):
```

用处最大的查询可能要算“Select the most recent backup for a client”（第 5 项选择）和“List Jobs where a given File is saved”（第 2 项选择）。对于那些麻烦的用户，他们老是记不住他们不小心删除的文件到底在哪儿，后一个选项就很方便。另一条作用大的命令是第 4 项选择“Enter SQL list command”，它让您输入格式正确的任何 SQL 查询。

假如某个用户需要恢复出来一份 2006 年 4 月前后的 **pw_expire.pl** 脚本的副本，但是这个用户却不能肯定那个时间自己正在用哪台机器做开发。另外，他喜欢把文件恢复到原来机器的**/tmp** 目录下。类似这样的要求会让许多系统管理员抓狂，但对于 Bacula 管理员却是小菜一碟。（遗憾的是，Bacula 的搜索结果格式太宽了，我们不得不把它缩小到下面这种几乎看不清的比例。）

```
* restore
To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of comma separated JobIds to select
...
     9: Cancel
Select item:  (1-9): 2
Enter Filename: pw_expire.pl
```

JobId	Client	Name	StartTime	JobType	JobFiles	JobBytes
4,667	harp	/home/seidel/pw_expire.pl	2006-05-09 23:55:21	B	176,072	12,664,064,296
5,220	bull	/home/seidel/bin/pw_expire.pl	2006-07-08 20:15:49	B	14	248,386
5,216	harp	/home/seidel/pw_expire.pl	2006-07-08 14:21:05	B	176,451	12,968,821,897
4,523	harp	/home/seidel/pw_expire.pl	2006-04-25 23:43:56	B	167,435	12,539,537,882

Bacula 列出备份里出现的若干个 **pw_expire.pl** 文件，反映出该文件在 2006 年 4 月的那个版本在客户机 harp 上，它是作为 4 523 作业的一部分备份的。Bacula 接着让用户返回到恢复菜单，在这个菜单里选择第 3 项（“Enter list of comma separated JobIds to select”），把注意力放在这个 4 523 作业上。

www.TopSage.com

```
Select item:  (1-9): 3
Enter JobId(s), comma separated, to restore: 4523
You have selected the following JobId: 4523
Building directory tree for JobId 4523 ...
1 Job inserted into the tree.
You are now entering file selection mode, where you add and
remove files to be restored. All files are initially added.[9]
Enter "done" to leave this mode.

cwd is: /
$ cd /home/seidel
cwd is: /home/seidel
$ dir
...
-rwxr-xr-x   1 seidel   atrust      321  2005-10-27 11:25:24  /home/seidel/pw_expire.pl
$ mark pw_expire.pl
1 files marked.
$ done
```

注意，即使 Bacula 原本显示的作业号（ID）有一个逗号（4,523），但是我们再次输入这个 ID 的时候，就要省略这个逗号；不然的话，Bacula 就会把 4,523 当作 4 和 523 两个作业 ID。

Bacula 载入特定的作业之后，它就会启动交互式的恢复模式，用户可以在这个模式下浏览找到这个文件，用 **mark** 命令选中它。标记完文件之后，用 **done** 命令退出交互模式。

```
Bootstrap records written to /var/bacula/working/restore.bsr
The restore job will require the following Volumes:

   003048L1

1 file selected to be restored.

Defined Clients:
     1: bull
     2: harp
Select the Client (1-2): 2
```

接下来，Bacula 写一个引导文件，它会用这个文件执行恢复工作、显示它所需要的磁带卷名，以及提示用户选择一个客户机——应该把文件恢复到这个客户机上。在本例中，我们把文件恢复到原来的主机 harp 上。

```
Run Restore job
JobName:    RestoreFiles
Bootstrap:  /var/Bacula/working/restore.bsr
Where:      /scratch02/restore/
Replace:    always
FileSet:    BullHomes
Client:     harp
Storage:    SureStore
When:       2006-07-08 16:45:20
Priority:   10
OK to run? (yes/mod/no): mod
```

对于本例提到的这个作业来说，我们修改了默认的设置。具体来讲，就是正确的文件集名为 harp，它是在控制器守护进程的配置文件里设置的默认文件集。此外，我们把这次恢复作业的位置按照用户的要求改为**/tmp**。

[9] *如果没有选择任何文件，却走到*这个恢复阶段的话，Bacula 会提示没有选过文件。

www.TopSage.com

```
Select parameter to modify (1-11): 9
Please enter path prefix for restore (/ for none): /tmp
Run Restore job
JobName:    RestoreFiles
Bootstrap:  /var/Bacula/working/restore.bsr
Where:      /tmp
Replace:    never
FileSet:    harp
Client:     harp
Storage:    SureStore
When:       2006-07-08 16:45:20
Priority:   10
OK to run? (yes/mod/no): yes
Run command submitted.
Restore command done.
```

改好之后，我们把作业提交给控制器执行。之后我们用 **messages** 命令观察作业的日志输出。

```
* messages
08-Jul-2006 17:06 bull-dir: Start Restore Job RestoreFiles.2006-07-08_17.06.02
08-Jul-2006 17:06 bull-sd: Ready to read from volume "003048L1" on device /dev/nst0.
08-Jul-2006 17:06 bull-sd: Forward spacing to file:block 11:0.
08-Jul-2006 17:08 bull-sd: End of Volume at file 11 on device /dev/nst0, Volume
    "003048L1"
08-Jul-2006 17:08 bull-sd: End of all volumes.
08-Jul-2006 17:08 harp: -rwxr-xr-x  1 seidel  atrust      321 2005-10-27 11:25:24
    /tmp/home/seidel/pw_expire.pl
08-Jul-2006 17:08 bull-dir: Bacula 1.34.6 (28Jul04): 08-Jul-2006 17:08
JobId:                  5217
Job:                    RestoreFiles.2006-07-08_17.06.02
Client:                 harp
Start time:             08-Jul-2006 17:06
End time:               08-Jul-2006 17:08
Files Expected:         1
Files Restored:         1
Bytes Restored:         321
Rate:                   0.0 KB/s
FD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:            Restore OK

08-Jul-2006 17:08 bull-dir: Begin pruning Jobs.
08-Jul-2006 17:08 bull-dir: No Jobs found to prune.
08-Jul-2006 17:08 bull-dir: Begin pruning Files.
08-Jul-2006 17:08 bull-dir: No Files found to prune.
08-Jul-2006 17:08 bull-dir: End auto prune.
```

9.8.13 监视和调试 Bacula 的配置

用户可以使用控制台的 **status** 命令查询各个 **Bacula** 的守护进程了解情况。下面的例子给出了有关控制器守护进程的信息。输出的内容包括后续作业、当前正在运行的作业、以及终止作业的信息。

```
* status dir
bull-dir Version: 1.34.6 (28 July 2004) i686-redhat-linux-gnu redhat Enterprise release
Daemon started 09-Jun-06 15:36, 269 Jobs run since started.
```

www.TopSage.com

```
Scheduled Jobs:
Level          Type      Scheduled        Name     Volume
===========================================================
Incremental    Backup    08-Jul-06 20:10  harp     003005L1

Running Jobs:
JobId    Level    Name                          Status
=====================================================
5216     Full     harp.2006-07-08_14.21.03      is running

Terminated Jobs:
JobId    Level    Files    Bytes          Status   Finished         Name
====================================================================
5205     Incr     204      898,066,578    OK       06-Jul-06 20:36  harp
```

Bacula 的作业能出一份作业报告，按照控制器守护进程的配置文件中作业的 Message 资源来发送。这份报告包括的信息有：所用的磁带卷、所备份文件的大小和数量，以及可能出现的任何错误。报告通常能够提供给用户充分的信息来做任何小问题的故障排查工作。

经常出现的两个问题是：客户机的文件守护进程没有运行，存储守护进程没有能找到任何可以追加的磁带卷。在下面的例子里，控制器守护进程报告说，备份作业因一个致命错误而中止，原因是它没办法和主机 harp 上的文件守护进程进行通信。在汇总报告的末尾还能再看到这个错误。

```
bull-dir: Start Backup JobId 5215, Job=harp.2006-07-08_13.19.49
bull-dir: harp.2006-07-08_13.19.49 Error: Bacula 1.34.6 (28Jul04): 08-Jul-2006 13:49

JobId:                  5215
Job:                    harp.2006-07-08_13.19.49
Backup Level:           Full
Client:                 harp
FileSet:                "harp" 2006-01-06 20:37:06
Start time:             08-Jul-2006 13:19
End time:               08-Jul-2006 13:49
FD Files Written:       0
SD Files Written:       0
FD Bytes Written:       0
SD Bytes Written:       0
Rate:                   0.0 KB/s
Software Compression:   None
Volume name(s):
Volume Session Id:      262
Volume Session Time:    1149888981
Last Volume Bytes:      0
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:
SD termination status:  Waiting on FD
Termination:            *** Backup Error ***

bull-dir: harp.2006-07-08_13.19.49 Warning: bnet.c:666 Could not connect to
    File daemon on 192.168.7.2:9102. ERR=Connection refused Retrying ...
bull-dir: harp.2006-07-08_13.19.49 Fatal error: bnet.c:672 Unable to connect to
    File daemon on 192.168.7.2:9102. ERR=Connection refused
```

下面例子里的存储守护进程报告说，指定的存储池里没有磁带卷来执行要求做的备份作业。要么向存储池增加一个新磁带卷，要么清空一卷磁带重新使用，就可以解决这个问题，不必重新开始这个作业，除非明确地取消它，否则 Bacula 应该继续执行它。

www.TopSage.com

```
bull-sd: Job baikal.2006-07-04_20.10.06 waiting. Cannot find any appendable volumes.
Please use the "label" command to create a new Volume for:
    Storage:      SureStoreDevice
    Media type:   LTO1
    Pool:         SSFullPool
```

如果需要更详细地知道正在执行的是什么守护进程，就可以在启动命令后面加上**-d***nnn* 选项，让这些守护进程向控制台发大量调试信息。例如：

```
$ sudo ./bacula start -d100
```

其中 *nnn* 代表调试的级别，一般它的值介于 50～200 之间。数字越大，显示的信息越多。用户还可以在控制台内用 **setdebug** 命令启动调试。

9.8.14 Bacula 的替换工具

从网上可以下载到其他一些免费或者共享的备份工具。下面是其中特别值得一提的工具，其他工具则仍然处于积极开发的状态中。

- Amanda：一种非常流行而且也经过考验的备份系统，它能把 UNIX 和 Linux 系统备份到一个磁带机上。参见 www.amanda.org。
- Mondo Rescue：一种可以把 Linux 系统备份到 CD-R、DVD-R、磁带或者硬盘上的工具。这一工具对于裸机恢复格外有用。到 www.mondorescue.org 多了解一下吧。
- **rsync**：一种免费工具，许多 Linux 默认都会安装它。它可以让文件在一台计算机到其他计算机做同步，可以配合 SSH 一起在 Internet 上安全地传输数据。参见 17.3.2 节了解对 **rsync** 的更多讨论。
- **star**：**tar** 的一种更快实现，我们所举的所有 Linux 发行版本中都有 **star**。

9.9 商用备份产品

我们都愿意认为 Linux 系统是世界上唯一的操作系统，但遗憾的是，实际情况并非如此。当了解商业的备份解决方案后，您应该考虑它们是否能处理任何别的操作系统，您要负责对这些操作系统做备份。大多数现代产品都会考虑跨平台问题，好让您在自己的 Linux 备份方案中包含 UNIX、Windows 和 Macintosh 工作站。您还必须考虑那些非 Linux 的存储阵列和文件服务器。

还应该通过备份保护用户的膝上机和其他那些不是总能连入网络的计算机，以防其发生故障。在考察商业产品时，您可能想问是不是每个产品都足够聪明，不会从每台膝上机中备份相同的文件。您实际需要多少个 **command.com** 的副本呢？

既然我们发现 Bacula 能够很好地为我们工作，所以我们对那些商业产品没有太多经验。我们向一些商业站点中的合作伙伴询问了他们对其所使用的系统的印象。他们的评论如下。

9.9.1 ADSM/TSM

该产品由 IBM 开发，后来被 Tivoli 收购。虽然这个产品又再次归 *IBM* 所有，但现在它是以 Tivoli Storage Manager（TSM，Tivoli 存储管理器）的名字在市场上销售的。TSM 是一个数据管理工具，它也能够处理备份。您可以在 www-306.ibm.com/software/tivoli 找到更多信息。

www.TopSage.com

大学网 TopSage.com

1. 优点：
- IBM 的产品，存在至今；
- 吸引人的定价和租用选择；
- 非常低的故障率；
- 使用磁盘高速缓存，这对于备份低速客户机很有用；
- 可以处理 Windows 客户机；
- 非常好的文档资料（单独标价）。

2. 缺点：
- GUI 界面设计差；
- 每两个文件＝数据库中 1K 的存储；
- 设计永远都在增加。

9.9.2 Veritas

2005 年，Veritas 同 Symantec 合并了。它们销售很多系统的备份解决方案。当您访问它们的 Web 站点（www.symantec.com）时，一定要选择适合于自己的产品。

1. 优点：
- 相当好的 GUI 界面；
- 可以直接和 Network Appliance 文件编档工具连接；
- 用于 Linux 系统的“推（push）”安装；
- 能够以 GNU **tar** 格式写磁带；
- 集中式的数据库，但也能够支持一个分布式的备份系统。

2. 缺点：
- 还存在一些缺陷；
- 定价混乱而且麻烦。

9.9.3 其他选择

W.Curtis Preston 建立了一个有关备份方面内容的网站（如磁盘镜像产品、高级文件系统产品、远程系统备份产品、离线数据控制产品等）。该站点的网址是 www.backupcentral.com。不清楚这些信息是否还在得到更新，但是现有的材料仍然相当有用。

9.10 推荐读物

PRESTON, W. CURTIS. *Backup & Recovery*. Sebastopol, CA: O'Reilly Media, 2006.

9.11 习题

E9.1 调查一下您的站点所采用的备份方案。哪一（几）台机器执行备份操作？使用什么类型的存储设备？磁带保存在哪里？对当前系统提出改进建议。

E9.2 在采用 Bacula 的系统上恢复文件需要哪些步骤？您怎样才能找到正确的磁带？

www.TopSage.com

★E9.3 给定了下面的 **df** 输出和**/etc/dumpdates** 文件，确定执行所要求的恢复操作需要的步骤。列举出您所作的假设。假定要求恢复的数据是 1 月 18 日的。

在 khaya.cs.colorado.edu 机器上，**df** 的输出为：

```
/dev/hda8       256194    81103    161863    33%  /
/dev/hda1        21929     4918     15879    24%  /boot
/dev/hda6      3571696    24336   3365924     1%  /local
/dev/hda10      131734     5797    119135     5%  /tmp
/dev/hda5      1815580  1113348    610004    65%  /usr
/dev/hda7       256194    17013    225953     7%  /var
```

khaya.cs.colorado.edu 机器上的**/etc/dumpdates** 文件：

```
/dev/hda8        2 Tue Jan  17  22:59:23 2006
/dev/hda6        3 Tue Jan  17  22:51:51 2006
/dev/hda7        3 Tue Jan  17  22:50:24 2006
/dev/hda5        9 Tue Jan  17  22:46:25 2006
/dev/hda5        1 Thu Jan  12  22:45:42 2006
/dev/hda7        0 Thu Jan  12  23:14:47 2006
/dev/hda6        1 Thu Jan  12  23:14:32 2006
/dev/hda8        1 Thu Jan  12  23:14:17 2006
/dev/hda6        0 Tue Jan  10  22:47:31 2006
/dev/hda1        1 Sun Jan   8  22:16:05 2006
/dev/hda7        1 Sat Jan   7  22:08:09 2006
/dev/hda1        4 Tue Jan   3  22:51:53 2006
/dev/hda7        2 Sat Dec  24  22:53:52 2005
/dev/hda5        0 Thu Nov   3  22:46:21 2005
/dev/hda1        0 Wed Sep  21  22:46:29 2005
/dev/hda8        0 Wed Aug  24  23:01:24 2005
/dev/hda1        3 Fri Jul  29  22:52:20 2005
/dev/hda6        2 Fri Jul  29  23:01:32 2005
```

a）“请恢复几天前某个时刻我的整个主目录（**/usr/home/clements**）。我似乎已经弄丢了我原来项目的整个代码库。”

b）“我不小心在我的机器 khaya 上执行了 **sudo rm –rf /***命令。请您从最近一次的备份恢复全部文件系统，好吗？”

c）“我上个月从 Aimster 收集的所有 MP3 文件都没了。它们保存在**/tmp/mp3/**里。您可以为我恢复它们吗？”

★E9.4 为下面的场合设计一个备份方案。假定每台计算机都有一个 100GB 的硬盘，用户的主（home）目录都在本地保存。选择一种备份设备，在成本和所需支持之间求得平衡，解释您的理由。列举出您所作的任何假设。

a）有 50 台机器的一家研究机构。每台机器保存有大量经常变动的重要数据。

b）有 10 台机器的一家小软件公司。保存源代码的中央服务器有 500GB 的硬盘空间。源代码整天都在不断变化。个人用户的主目录不会经常变化。对成本几乎不作考虑，但安全极为重要。

c）有两台计算机的家庭网络。成本是最重要的考虑因素。

★E9.5 为上述 3 种场合分别设计恢复策略。

★E9.6 列出您通过一个安全的 SSH 通道执行安全 **rdump** 操作所采取的步骤。

★★E9.7 编写 Bacula 的配置语句，实现您为习题 9.4 所做的备份计划。

www.TopSage.com

第 10 章　系统日志与日志文件



系统守护进程、内核和各种工具都会产生一些数据，这些数据被记录下来，最终保存在容量有限的硬盘上。其中大部分数据有用的寿命都是有限的，因此需要对它们进行汇总、压缩、存档并且最终扔掉。

10.1　日志记录的策略

日志记录的策略随站点不同而不同。常见的方案包括以下几种：

- 立刻扔掉所有数据；
- 定期重新设置日志文件；
- 轮换日志文件，把数据保留一定的时间；
- 将日志压缩并保存到磁带或其他永久性介质上。

对于您的站点来说，正确的选择取决于磁盘空间有多大，您的安全意识有多强。即使具有充足的磁盘空间，还是不要让日志文件过快增长。

无论选择何种方案，都必须用 **cron** 自动维护日志文件。有关该守护进程的详细内容，请参考第 8 章。

10.1.1　扔掉日志文件

我们并不建议扔掉所有的日志信息。遭受到安全问题的站点常常会发现，日志文件提供了非法入侵的重要证据。日志文件还有助于提醒您有关硬件和软件方面的问题。总而言之，如果磁盘空间足够的话，建议至少保存数据一个月，然后才可以丢弃。在实际生活中，您可能需要很长一段时间才会意识到站点已经被黑客攻破，这时就需要查看以前的日志。如果有必要回顾更远的过去所发生的事件，那么可以从备份磁带上恢复较早的日志文件。

www.TopSage.com

有些管理员听任日志文件任意增长，直到它们变成麻烦后才将其清除并从零开始。这种方案比一点儿数据都不保存要好，但这样做无法保证日志记录项能够存留任何特定的时间长度。磁盘的平均利用率也要比使用其他管理方案的利用率高。

在很个别的情况下，一个站点决定保留某些日志文件，可能不是为了任何有益的目的，而更多是为了应付传票。处于这种境地的一个站点可能会留着几周的日志数据，但是它或许要保证这些文件一定不能被存到永久性的介质上。有一个这样的案例：微软已经不止一次受到指控，其管理日志文件和电子邮件的政策破坏性过强。原告称，微软的数据保留政策无异于毁掉证据，尽管删除行动（或者至少是删除策略）是在法律诉讼之前就已经有了。遗憾的是，目前断定法庭将最终如何回应这些诉讼还为时过早[1]。在美国，Sarbanes-Oxley 法案最近规定了保留记录的新要求，参考 30.12.8 节。

10.1.2 轮换日志文件

把每周或者每月的日志信息保存在一个单独的文件里，这是一种通常（但不是通行）的做法。这些周期性的文件都会保留特定一段时间，然后被删除。我们专门在一台中央日志主机上划了一个硬盘分区（**/var/log**）给日志文件。

在每个轮换周期结束的时候，有一个脚本或者工具程序更改每个文件的名字，然后把较早的数据向文件链的结尾推。例如，假设某个日志文件的名字叫做 **logfile**，则它的备份文件可能叫做 **logfile.1**、**logfile.2**，依此类推。如果每周轮换一次，并且保存 8 周的数据，那么就会有一个 **logfile.8** 文件但没有 **logfile.9** 文件。每周随着 **logfile.7** 文件覆盖 **logfile.8** 文件，**logfile.8** 中原来的数据就没了。

稍微麻烦一点，压缩一下数据就能延长保存的时间。您可以运行 **zgrep**，不必解压压缩文件，就能搜索压缩文件。

假定每周都要留意某个文件，而且（为了简化例子）假定希望存档该文件三周的内容。下面的脚本就可以实现一个适当的轮换策略：

```
#!/bin/sh
cd /var/log
mv logfile.2 logfile.3
mv logfile.1 logfile.2
mv logfile logfile.1
cat /dev/null > logfile
chmod 600 logfile
```

对于某些日志文件来说，权属信息很重要。您可能需要以日志文件属主的身份，而不是以 root 的身份从 **cron** 运行自己的轮换脚本，或者要在这些命令中加上 **chown** 命令。

大多数 Linux 发行版本（包括我们全部的示例版本）都支持一种叫做 **logrotate** 的很不错的日志轮换工具，我们将从 10.3 节开始介绍它。比起编写您自己的脚本来说，它要容易得多（也可靠得多），如果您的发行版本没有包含它，那么还是值得找到并安装它的。

某些站点用日期而不是序列号来标识日志文件，例如 **logfile.tues** 或 **logfile.2005.04.01**。这种系统实现起来稍微困难一点儿，但如果需要频繁用到旧的日志文件，那么还是值得这么做的。在 **Perl** 里设置用日期给文件起名字要比在 **sh** 中容易得多。一种不需要任何编程技巧的习惯用法是：

```
mv  logfile  logfile.`date  +%Y.%m.%d`
```

这种方法也有能让 **ls** 命令对日志文件按时间顺序进行排序这一优点。（**ls** 的**-t** 选项让它把任何目

[1] 值得一提的是，在出现官司苗头的时候（或者更为恶劣一些，在诉讼过程期间）就删除相关的电子邮件或者日志文件的做法注定不会占理。只有正在执行的明确不保留数据的策略才会有望得到法律保护。

www.TopSage.com

录下的文件按修改时间排序，但是如果这些文件没有要求就这样自行安排也挺不错。）

有些守护进程的日志文件自始至终都处于打开状态。由于文件系统工作方式的缘故，我们举例的脚本不能用于这样的守护进程。日志记录数据不会保存到再次创建的 **logfile** 中，而是继续保存到 **logfile.1** 里；即使在将文件重新命名之后，对原来文件的活动引用仍然存在。若想安装一个新的日志文件，则必须向守护进程发信号或者杀死并重启它。参考本书的相关章节（或者您的手册）确定在每种情况必须采取什么样的步骤。

下面的例子在上述例子的基础上进行了修改，它同时使用了压缩和信号：

```
#!/bin/sh
cd /var/log
mv logfile.2.gz logfile.3.gz
mv logfile.1.gz logfile.2.gz
mv logfile logfile.1
cat /dev/null > logfile
chmod 600 logfile
kill -signal pid
gzip logfile.1
```

其中的 *signal* 代表向程序发出的写日志文件的适当信号，*pid* 是它的进程 ID。该信号可以直接写入脚本，但必须动态判断守护进程的 PID，这可以通过读取该守护进程留下的文件（比如**/var/run/syslogd.pid**，这将在下面给予介绍）或是使用 **kill** 的变体 **killall** 来实现，**killall** 能够在进程表中查找 PID。例如，下面的命令

```
killall -e -HUP syslogd
```

等价于

```
kill -HUP `cat /var/run/syslogd.pid`
```

10.1.3 存档日志文件

除非您就是希望避免留下书面记录，否则应该在常规的系列备份中包含有日志文件。因为它们包含的信息对于调查安全事故来说至关重要，所以在您的转储频度允许的情况下，要以最高频度备份它们。日志文件变化很频繁，所以它们能够代表要保存在增量备份上的系统信息的重要部分。在设计您的日志政策和您的备份策略时要记得它们两者之间的相互作用关系。有关备份的更多信息，可参见第 9 章。

除了作为例行备份的一部分之外，日志也可以存档到另一套单独的磁带上。相比之下，单独的一套磁带使用起来不方便，但它们带来的建档负担要轻，同时又不影响您循环使用转储磁带的能力。在使用单独的磁带时，我们建议您采用 **tar** 格式，并编写一个脚本来自动执行备份方案。

10.2 Linux 的日志文件

传统的 UNIX 系统往往因为它们做日志的方法不一致，甚至还有点儿怪异而受到批评。幸运的是，虽然 Linux 系统的每一种发行版本都有自己的一套给日志文件命名和分类的方法，但是一般而言 Linux 要好一点儿。Linux 的软件包大多将它们的日志信息记录到**/var/log** 目录下的文件里。在有些发行版本上，个别日志也保存在**/var/adm** 里。

www.TopSage.com

现如今，大多数程序都把它们的日志项发到一个称为 syslog 的中央清理系统，syslog 将在本章后面予以介绍。默认的 syslog 配置一般将这些信息的大部分转储到**/var/log** 中的某个地方。请检查系统日志的配置文件**/etc/syslog.conf**，找出这些信息究竟存在什么地方。有关 **syslog.conf** 文件的格式，请参考 10.4 节。

表 10.1 总结了一些在我们举例的系统中较为常见的日志文件的有关信息。具体有如下几项：

- 存档、汇总或截断的日志文件；
- 创建各个日志文件的程序；
- 有关如何指定每个文件名的指示；
- 我们认为留意或者清理的合理频率；
- 使用日志文件的发行版本（在我们的例子中出现过）；
- 文件内容的说明。

除非特别说明，否则文件名都是相对于 **var/log** 而言的。

“出处”一列中的字符说明了日志文件的指定方法：S 代表使用 syslog 的程序，F 代表使用一个配置文件的程序，而 H 则代表文件名是否直接包含在代码中。“频率”一列表明了我们建议的清理频率。“发行版本”一列给出了该项适用的发行版本。

虽然各个发行版本上日志文件的所属关系和模式都有不同，但日志文件一般由 root 所有。就我们看来，大多数日志的模式都应该是 600（只有属主有读写权限），因为它们的内容对于黑客来说可能有帮助。如果用户的水平相当高，那么能够查看日志对用户会有好处。在这种情况下，放开日志文件的部分权限也是合理的。

专为日志建立一个用户组，把日志文件的属组设为这个用户组，让这个用户组的成员都能读取日志文件，这是另一种合理的做法。您可以把本地的系统管理员加入到这个用户组里，让他们不必使用 **sudo** 命令就能看到日志文件的内容。如果站点有一个初级系统管理员，他没有完全的 **sudo** 特权，这样的安排就特有用。

至少，应该限制用户对 **secure**、**auth.log** 和 **sudo.log** 文件不经意地查看。绝对不要把任何日志文件的写权限交给属主之外的任何人。

值得说明的一点是，表 10.1 中给出的日志文件大多数都是由 syslog 维护的，但是 syslog 的默认配置则随系统的不同而变化很大。采用了比较一致的**/etc/syslog.conf** 文件后，各 Linux 发行版本上的日志文件看上去就相当相似了。

表 10.1 日志文件列表

文 件	程 序	出 处[a]	频 率[a]	发行版本[a]	内 容
auth.log	**su** 等[b]	S	M	DU	授权
apache2/*	**httpd** 第 2 版	F	D	SDU	Apache HTTP 服务器的日志（第 2 版）
boot.log	**rc** 脚本	F[c]	M	RF	系统启动脚本的输出
boot.msg	内核	H	-	S	内核消息缓冲的转储
cron	**cron**	S	W	RF	**cron** 的执行情况和出错信息
cups/*	CUPS	F	W	所有	与打印有关的消息（CUPS）
daemon.log	许多	S	W	DU	所有守护进程功能的消息
debug	许多	S	D	DU	调试输出
dmesg	内核	H	-	RFDU	内核消息缓冲的转储
dpkg.log	**dkpg**	F	M	DU	软件包管理日志
faillog	**login**	H	W	SDU	不成功的登录企图
httpd/logs/*	**httpd**	F	D	RF	Apache HTTP 服务器的日志（在**/etc** 下）

www.TopSage.com

续表

文 件	程 序	出 处[a]	频 率[a]	发行版本[a]	内 容
kern.log	内核	S	W	DU	所有 kern 功能的消息
lastlog	**login**	H	-	所有	每个用户上次登录的时间（二进制）
mail*	与 mail 有关的	S	W	所有	所有 mail 功能的消息
messages	许多	S	W	所有	经常是主要的系统日志文件
rpmpkgs	**cron.daily/rpm**	H	D	RF	已安装的 RPM 软件包清单
samba/*	**smbd** 等	F	W	-	Samba（Windows/CIFS 文件共享）
secure	**sshd**、**sudo** 等	S	M	RF	保密的授权消息
syslog	许多	S	W	DU	经常是主要的系统日志文件
warn	许多	S	W	S	所有的警告/出错级的消息
wtmp	**login**	H	M	所有	登录记录（二进制）
Xrog.*n*.log	**Xorg**	F	W	RFS	X 窗口服务器的出错信息
yum.log	**yum**	F	W	RF	软件包管理日志

a. 出处：S=syslog, H=硬连接，F=配置文件。
频率：D=每天，W=每周，M=每月。
发行版本： R=Red Hat Enterprise Linux，F=Fedora，D=Debian，S=SUSE，U=Ubuntu。
b. **passwd**、**login** 和 **shutdown** 也可以写入授权日志。在 Red Hat 和 Fedora 系统上，它在**/var/adm** 里。
c. 实际是通过 syslog 做日志，但是在**/etc/initlog.conf** 中配置功能和级别。

10.2.1 特殊的日志文件

大多数日志是文本文件，当发生“感兴趣的”事件时，就会向这些文件中写入日志记录行。但是表 10.1 中列出的几个日志的来龙去脉却非常不同。

/var/log/wtmp 中包含用户登录系统和退出系统的记录，也包含了表明系统何时重启或者关机的记录项。因为新的记录项只是简单地追加到文件的末尾，所以它是一种相当普通的日志文件。不过，**wtmp** 文件是以二进制形式保存的。使用 **last** 命令可以解读这些信息。尽管 **wtmp** 的格式不一般，但是应该像其他任何日志文件那样，轮换或者截断这个文件，因为它的自然特性是无限制的增长。

/var/log/lastlog 包含的信息类似于**/var/log/wtmp** 中的信息，但是它只记录每个用户上次登录的时间。它是一个稀疏的二进制文件，以 UID 作为索引。如果您的 UID 是以某种数字序列来指定的，那么虽然这肯定对现实世界没什么影响，但这个文件会变得小一些。**lastlog** 文件不需要轮换，因为除非有新用户加入，否则它的大小保持不变。参见 9.3.1 节的脚注了解有关“稀疏”文件的更多知识。

10.2.2 内核和启动日志

内核和系统启动脚本反映出了在日志领域内的一些特殊的挑战。对于内核来说，问题在于既要创建有关引导进程和内核操作的永久记录，又不能增加对任何特殊的文件系统或者文件系统组成的依赖性。对于启动脚本而言，挑战在于既要捕捉启动过程的连贯描述，又不能总是试图把任何系统守护进程都和启动日志文件扯到一块，干扰任何程序自己的日志机制，或是让启动脚本做两份日志项或者把日志输出重定向。

内核的日志机制是通过让内核把它的日志项保存在一个大小有限的内部缓冲区来做到的。缓冲区要足够大，以便能放得下内核在引导时的活动所产生的全部消息。一旦系统全部启动以后，用户进程

www.TopSage.com

就可以访问内核的日志缓冲，最终处理它的内容。各发行版本一般是运行 **dmesg** 命令，并把它的输出重定向到**/var/log/dmesg**（RHEL、Fedora、Debian 和 Ubuntu）或者**/var/log/boot.msg**（SUSE）。这是查看最近启动过程信息的最好的地方。

内核当前运行的日志机制是由一个叫做 **klogd** 的守护进程处理的。**klogd** 的功能实际上是 **dmesg** 功能的超集，除了转储内核的日志并退出之外，它还可以在内核缓冲区内的消息产生的时候读取它们，并将其发送到一个文件或者 syslog。在正常的运行方式下，**klogd** 采取后一种模式。syslog 根据对“kern”的指令来处理这些消息（一般把它们发送到**/var/log/messages**）。

我们举例用的发行版本的启动脚本在一开始转储日志消息的时候，都不用 **dmesg** 的**-c** 标志，所以虽然读取了内核的消息缓冲区，但并不清除它。当 **klogd** 启动的时候，它在缓冲区里发现了和 **dmesg** 见到的一样的一组消息，于是把它们发送给 **syslog**。出于这个原因，有些日志项既会出现在 **dmesg** 或者 **boot.msg** 文件里，又会出现在如**/var/log/messages** 这样由 **syslog** 管理的文件里。

内核日志机制的另一个问题是要在系统控制台做适当的管理。伴随系统的引导，在控制台上输出信息是很重要的。不过，一旦系统已经启动了，控制台的消息可能与其说是帮助，不如说是烦人，如果用控制台登录的话就更是这样了。

dmesg 和 **klogd** 都可以让您用一个命令行标志设置内核的控制台日志级别。例如：

```
$ sudo dmesg-n 2
```

7 级提供的信息最多，还包括调试信息。1 级只包含内核的“panic（恐慌）”消息（编号小的级别最严重）。所有的内核消息继续进入中央缓冲区（也进入 syslog），而不管它们是否被转发到控制台。

内核在**/proc/sys** 目录下提供了一些控制文件，让大量重复发生的日志消息在来源处就被阻塞。参见 28.4 节，了解通过设置哪些内核参数来实现一般控制机制的更多信息。这些专门的控制文件是**/proc/sys/kernel/printk_ratelimit** 和**/proc/sys/kernel/printk_ratelimit_burst**，前者指定在启动阻塞之后内核消息之间必须间隔最少多少秒（默认为 5s），后者规定在启动阻塞之前允许多少组消息通过（默认为 10）。这两个参数都是建议性的，所以它们不会绝对保证能制止大量消息。



遗憾的是，系统启动脚本的日志机制没有内核的日志机制管理得好。Red Hat Enterprise Linux 采用一条叫做 **initlog** 的命令来捕获启动命令的输出，并把它发送给 syslog。遗憾的是，无论什么时候运行命令，都必须明确调用 **initlog**，所以确实要以某种复杂性为代价才能捕获到信息。这些日志项最终还是会进入**/var/log/boot.log**。

我们举例用的其他系统都没有连续捕获启动脚本的历史输出。个别命令和守护进程能记录下来一些信息，但是大多数信息都不作日志记录。



Fedora 以前使用和 Red Hat 一样的 **initlog** 系统，但是现在这条提交日志记录的命令已经从启动脚本中注释掉了。幸好还有一个工具函数的中央库**/etc/init.d/functions**，您可以在那里去掉 **initlog** 的注释行重新启用它们。

10.3 logrotate: 管理日志文件

Erik Troan 提供了一种优秀的工具 **logrotate**，它实现了多种多样的日志管理策略，而且在我们举例的所有发行版本上都是标准应用。

logrotate 的配置文件由一系列规范组成，它们说明了要管理的日志文件组。出现在日志文件规范之外的选项（诸如下面例子中的 `errors`、`rotate` 和 `weekly`）适用于后面所有的规范，它们可以

www.TopSage.com

在特定日志文件的规范内被覆盖，也可以在文件的后面重新指定，修改默认值。

下面是一个有点儿故意人为的例子，它设定了几种不同的日志文件：

```
# Global options
errors errors@book.admin.com
rotate 5
weekly

# Logfile rotation definitions and options
/var/log/messages {
        postrotate
                /bin/kill -HUP `cat /var/run/syslogd.pid`
        endscript
}

/var/log/samba/*.log {
        notifempty
        copytruncate
        sharedscripts
        postrotate
                /bin/kill -HUP `cat /var/lock/samba/*.pid`
        endscript
}
```

这一配置每周轮换一次**/var/log/messages** 文件。它保持了这个文件的 5 个版本，并且在每次重置文件的时候通知 **syslogd**。Samba 的日志文件（这样的文件可能有好几个）也每周轮换一次，但不是移到一边后开始一个新的，而是复制以后截短。只有在轮换了所有的日志文件之后，才向守护进程 Samba 发 HUP 信号。

表 10.2 列出了 **logrotate.conf** 最有用的选项。

表 10.2　　logrotate 的选项

选　项	含　义
compress	压缩日志文件的所有非当前版本
daily, weekly, monthly	以指定的时间安排来轮换日志文件
delaycompress	压缩除了当前和最近之外的所有其他版本
endscript	标记 prerotate 或者 postrotate 脚本结束
errors *emailaddr*	向指定的 *emailaddr* 发送出错通知邮件
missingok	如果日志不存在，不会发出抱怨
notifempty	如果日志为空，则不轮换它
olddir *dir*	指定要放入 *dir* 里的日志文件老版本
postrotate	引入在轮换过日志之后要运行的脚本
prerotate	引入在进行任何改动之前运行的脚本
rotate *n*	在轮换方案中包括 *n* 个版本的日志
sharedscripts	只为整个日志组运行一次的脚本
size=*logsize*	如果日志文件大于 *logsize*（例如，100K，4M）才轮换

logrotate 一般每天由 **cron** 运行一次。它的标准配置文件是**/etc/logrotate.conf**，但是在 **logrotate** 的命令行上一次可以出现多个配置文件（或者包含配置文件的目录）。这一特性被我们所举的发行版本充分利用，它们规定**/etc/logrotate.d** 目录是保存 **logrotate** 配置文件的标准位置。懂得 **logrotate** 的软件包（这种软件有很多）能把执行日志管理指令作为它们安装过程的一部分，这大大地简化了管理工作。

www.TopSage.com

除了 **logrotate** 之外，Debian 和 Ubuntu 还提供了一种更简单的程序，它叫做 **savelog**，能够管理单个文件的轮转工作。它比 **logrotate** 更直接，不用（或者不需要用）配置文件。有些软件包更愿意使用它们自己的 **savelog** 配置，而不用 **logrotate**。

10.4 syslog：系统事件的日志程序

syslog 最初是由 Eric Allman 编写的，它是一个综合的日志记录系统。syslog 有两项重要的功能：使程序员从繁重、机械的编写日志文件代码的工作中解脱来，使管理员更好地控制日志的记录过程。在出现 syslog 之前，每个程序都有自己的日志记录策略。系统管理员对保存什么信息或是信息保存在哪儿没有控制权。

syslog 相当灵活，它能让消息按照其来源和重要性（“严重性级别”）来排序，也能把消息送到各种目的地：日志文件、用户的终端、或者甚至是别的计算机。syslog 集中管理网络日志的功能是它最有价值的特性之一。

10.4.1 syslog 的其他替代方案



虽然 syslog 已经长期主导 UNIX 和 Linux 的日志系统，但是人们已经开发出了替代它的几种方案，试图解决它的一些缺点。syslog-ng（下一代 syslog）就是这其中之一，它现在成为 SUSE 系统上默认采用的日志工具。从配置的观点来看，它和标准的 syslog 有很大的不同，我们在本书里就不详细介绍它了。如果想在非 SUSE 的系统上试试，可以从 www.balabit.com 获得它。

syslog-ng 增加了更多的配置功能、能基于消息内容进行过滤，确保了消息一致性，以及在消息通过网络转发时针对防火墙的限制提供了更好的支持。

（圣地亚哥超级计算中心的）SDSC Secure Syslog 也叫做高性能 syslog。它通过实现 RFC 3195 的规范，提供了一种“达到举证水平”的审计系统。它的设计考虑到了大流量站点的需要，包含了大量性能优化措施。您可以从 sourceforge.net/projects/sdscsyslog 获得它的源代码。

10.4.2 syslog 的体系结构

syslog 包括 3 个部分：

- **syslogd**，日志守护进程（及其配置文件/etc/syslog.conf）；
- **openlog** 等，将消息提交给 syslogd 的库例程；
- **logger**，一条从 shell 提交日志记录项的用户级命令。

在下面的讨论中，我们先介绍 **syslogd** 的配置，然后简要说明如何从 **Perl** 脚本中使用 syslog。

syslogd 在系统引导时启动并连续运行，它不能用 **inetd** 来管理。懂得 syslog 的程序把日志项（使用 **syslog** 库例程）写到专门的文件**/dev/log** 中，这是一个 UNIX 的域套接口。**syslogd** 从这个文件中读取消息，参考其配置文件，并将各条信息分发到合适的目的地。

挂起信号（HUP，信号 1）可以让 **syslogd** 关闭它的日志文件，重新读取它的配置文件，并再度开始日志记录。如果用户修改了**/etc/syslog.conf**，那么必须向 **syslogd** 发送一个挂起信号来使修改生效。TERM 信号将使 **syslogd** 退出。

syslogd 将它的进程 ID 写入到文件**/var/run/syslogd.pid** 里。这一约定使得从一个脚本向 **syslogd**

www.TopSage.com

大家网 TopSage.com

发送信号变得比较容易。例如，下面的命令将发送一个挂起信号：

```
# kill-HUP `/bin/cat /var/run/syslogd.pid`
```

试图压缩或者修改一个已经由 **syslogd** 打开以供写入信息的日志文件是不太安全的，可能会导致不可知的结果。恰当的步骤是把老的日志移到别处，重新创建日志，权属关系和权限都保持不变，然后向 **syslogd** 发送一个 HUP 信号。这个过程用 **logrotate** 很容易就能实现，参考 10.3 节的例子。

10.4.3 配置 syslogd

配置文件**/etc/syslog.conf** 控制着 **syslogd** 的行为。这是一个格式相当简单的文本文件。空白行和以字符“#”开始的行都被忽略。基本格式为：

```
selector <Tab> action
```

例如，下面的一行：

```
mail.info       /var/log/maillog
```

将把来自电子邮件系统的信息保存在文件**/var/log/maillog** 中。

selector 用语法 *facility.level* 指明正在发送日志消息的程序（即“设备[facility]”）和消息的严重性级别。设备的名称和严重性级别必须从已定义值的一个简短列表中选择，程序不能使用自己定义的值。设备分别为内核、常用应用程序组、以及本地编写的程序进行定义。任何其他程序则归为普通设备“用户（user）”一类。

selector 可以包含特殊的关键字*和 none，其含义分别为“所有的”或“什么都没有”。*selector* 可以包括多个用逗号分开的设备。多个 *selector* 可以用分号组合起来。

一般而言，*selector* 相互之间为“或（OR）”的关系，和某个 *selector* 匹配的消息将由同一行的 *action* 处理。但是，不管同一行中其他的 *selector* 是如何定义的，只要有一个带有 none 级别的 *selector* 就会排除列出的设备。

这里有几个格式化和组合 *selector* 的例子：

```
facility.level                  action
facility1,facility2.level       action
facility1.level1;facility2.level2 action
*.level                         action
*.level;badfacility.none        action
```

表 10.3 列出了有效的设备名。目前的设备有 21 种。

syslogd 本身可以产生时间戳消息，如果 **syslog.conf** 中出现了“mark”设备（facility）为其指定的一个目的地，那么会记录这些消息。时间戳可以帮助用户确定机器是在早上 3:00 和 3:20 之间崩溃的，而不仅仅是“昨天晚上的某个时间”。这个信息在调试有规律发生的问题时有很大用处。例如，许多站点都经历过当清洁工在深夜插上吸尘器电源时发生跳闸而导致神秘崩溃的事。

表 10.3　　syslog 的设备名

设备（facility）	使用该设备的程序
*	除了“mark”之外的所有设备
auth	与安全和授权有关的命令
authpriv	敏感/保密的授权消息[a]
cron	守护进程 **cron**

续表

设备（facility）	使用该设备的程序
daemon	系统守护进程
ftp	FTP 守护进程，**ftpd**
kern	内核
local0-7	本地消息的 8 种类型
lpr	行式打印机的假脱机系统
mail	**sendmail** 以及其他与电子邮件相关的软件
mark	定期产生的时间戳
news	Usenet 新闻系统
syslog	**syslogd** 内部消息
user	用户进程（如果没有指定，这将是默认值）
uucp	为 UUCP 保留，并未使用

a. 所有与授权有关的消息都敏感。authpriv 消息和 auth 消息不应该让任何人都能读取。

如果系统十分繁忙，其他日志消息通常可以提供充分的时间戳信息。但在凌晨的几个小时并不总是这种情况。

syslog 的严重性级别按照重要性递减的顺序在表 10.4 中列出。

消息的严重性级别指明了它的重要性。各个级别间的区别有时候显得比较模糊。在 notice 和 warning 之间（以及在 warning 和 err 之间）有一条清晰的界限，但是 alert 和 crit 相比，在所表达的含义上准确的细微差别则有点要靠猜测。表 10.7 列出了各种常用软件程序采用的特定级别。

表 10.4 syslog 的严重性级别（降序）

级 别	大致含义	级 别	大致含义
emerg	恐慌状态	warning	警告消息
alert	紧急状态	notice	需要调查的事项
crit	临界状态	info	提供信息的消息
err	其他错误情况	debug	仅供调试

在 **syslog.conf** 文件中，这些级别指定了最低程度的重要性，只有重要性大于（包括）它的消息才会被记录下来。例如，来自电子邮件系统的一条警告级消息和 `mail.warning` 匹配，也同样和 `mail.notice`、`mail.info`、`mail.debug`、`*.warning`、`*.notice`、`*.info` 和`*.debug` 匹配。如果 **syslog.conf** 文件指定了 `mail.info` 消息应该被记录到某个日志文件中，那么 `mail.warning` 消息也将被记录到该文件中。

Linux 版的 syslog 对基本语法做了改进，它也允许在优先级前面加上字符=和!，分别表示“仅此优先级”和“除此优先级及其以上级别”的意思。表 10.5 给出了一些例子。

表 10.5 在 syslog.conf 中限定优先级的例子

选 择 符	含 义
`mail.info`	选择与邮件相关的、优先级为 info 或者更高的消息
`mail.=info`	只选择优先级为 info 的消息
`mail.info;mail.!err`	只选择优先级为 info、notice 和 warning 的消息
`mail.debug;mail.!=warning`	选择除 warning 之外的所有优先级

action 字段说明如何处理一则消息。表 10.6 列出了可以选择的处理方法。

www.TopSage.com

表 10.6 syslog 的动作

动作（Action）	含 义
filename	把消息写入本地机器上的一个文件里
@hostname	把消息转发给主机 *hostname* 上的 **syslogd**
@ipaddress	把消息转发给 IP 地址为 *ipaddress* 的主机
\|fifoname	把消息写入有名管道 *fifoname*[a]
user1,user2,...	如果用户登录到了系统上，那么就把消息写在用户的屏幕上
*	把消息写给目前已经登录的所有用户

a. 参考 **info mkfifo** 了解更多的信息。

如果采取了 *filename*（或者 *fifoname*）这样的动作，那么文件名应该用绝对路径。如果您指定了一个不存在的文件名，那么在首次有消息送到这个文件的时候，**syslod** 会创建它[2]。您可以在 *filename* 动作之前加一个短划线“-”，表示在写完每条日志记录项后，不应该对文件系统执行 **sync**。执行 **sync** 有助于在发生崩溃的时候尽可能多地保留日志信息，但是从磁盘吞吐上来看，代价又太大。

如果用的是 *hostname* 而不是 IP 地址，那么主机名必须能通过 DNS 或者 NIS 这样的转换机制解析出来。

虽然一个 selector 可以包含多个设备（facility）和级别，但是却不能提供多个动作。为了把一则消息发送到两个地方（比如，到本地文件和到中央日志主机），您可以在配置文件中包含具有相同 Selector 的两行。

因为 syslog 消息能够用来实施拒绝服务攻击，所以除非 **syslogd** 以**-r** 标志启动，否则它不会接受来自其他机器的日志消息。默认情况下，**syslogd** 也会拒绝充当第三方消息转发器，从一台网络主机来的消息不会再被发给另一台网络主机。使用**-h** 标志能够取消这一功能。（如果您想让这些选项始终都被启用，那么对于 RHEL 和 Fedora，可以在脚本**/etc/rc.d/init.d/syslog** 中加上这些标志，对于 Debian 和 Ubuntu，可以在脚本**/etc/init.d/sysklogd** 里加上这些标志。）

10.4.4 设计站点日志方案

配置日志记录以便把重要的系统错误和警告保存在每台计算机上的一个文件中，在一个小型站点中这样做足够了，就像前面我们用 syslog 做的那样。**syslog.conf** 文件可以针对每台主机进行定制。

在一个大型网络中，必须有中央日志记录。它使得大量信息保持在可管理的状态，而且运气好的话，破坏网络计算机安全的人可能无法访问到审计数据。黑客常常通过修改系统日志来擦去他们留下的痕迹。如果日志消息刚一产生就被转移到别的位置，那么想要破坏它就更难了。站点级的防火墙不应该让外部站点向 **syslogd** 提交消息。

但要清楚一点，任何人都可以调用 **syslog**，并且可以伪造从任何守护进程或者应用程序来的日志项。syslog 也使用 UDP 协议，而这个协议并不保证是可靠的，消息可能会丢失。

选择一台稳定的计算机作为日志记录服务器，最好是有很好的安全措施而且登录用户不多的一台机器。其他计算机可以使用保存在中央主机上的一个通用配置文件。因此，只需要维护两个版本的 **syslog.conf** 文件。这样可以保持日志记录的完整性，但同时对于管理员来说又不再是个恶梦。有关在网络上分布文件的更多信息，请参见第 17 章。

为了达到最好的安全性，syslog 服务器应该通过防火墙与网络的其余部分隔离开，只能让连到 syslog 端口的网络连接通过，另外只能让得到允许可以向 syslog 记录日志的主机访问 syslog 服务器。根据周边的风险程度不同，可以从系统管理员的工作站用 SSH 连接，让管理员更方便地检查日志。

[2] 注意：这一行同 syslog 原本的实现正好相反，syslog 原来要求事先就创建好日志文件。

www.TopSage.com

某些非常大的站点可能还需要给日志记录体系添加几个级别。遗憾的是，syslog 的当前版本只能为最后一跳（hop）保留原来的主机名称。如果主机“client”将某些日志项发送到主机“server”，“server”又将它们发送到“master”，“master”只能知道消息来自“server”，而不知道是来自“client”。

10.4.5 配置文件举例

在 syslog 的配置上，不同的 Linux 发行版本变化非常大。因为读懂 **syslog.conf** 文件并不困难，所以就不详细看我们举例的发行版本中使用的配置文件了，它们都很直观易懂。我们转而要看一些常见的方法，如果您做出了不同于系统默认的选择，那么可能想要以这些方法来设置日志功能。

下面有 3 个 **syslog.conf** 文件的实例，分别对应于小型网络上的一台单机、较大网络中的一台客户机和一台中央日志记录主机。中央日志记录主机称为“netloghost”[3]。

一、单机

下面是单机的基本配置：

```
# syslog.conf file for small network or stand-alone machines

# emergencies: tell everyone who is logged on
*.emerg                                 *
#  important messages
*.warning;daemon,auth.info;user.none /var/log/messages
#  printer errors
lpr.debug                               /var/log/lpd-errs
```

第 1 个非注释行把紧急（emergency）消息写到所有当前在线用户的屏幕上。在系统即将关闭时，**shutdown** 所产生的消息就是一个紧急级别消息的例子。

第 2 行把重要（important）消息写入**/var/log/messages** 文件。info 级别比 warning 级别低，因此 `daemon,auth.info` 子句又包含了来自 **passwd**、**su** 和守护进程的日志记录消息。第三行把打印机错误信息写入到**/var/log/lpd-errs**。

二、网络客户机

网络客户机通常把重要的消息转发给中央日志记录主机：

```
# syslog.conf file for nonmaster machines

# Emergencies: tell everyone who is logged on
*.emerg;user.none                       *

# Forward important messages to the central logger
*.warning;lpr,local1.none               @netloghost
daemon,auth.info                        @netloghost

# Send some local stuff to the central logger too
local2.info;local7.debug                @netloghost

# Keep printer errors local
lpr.debug                               /var/log/lpd-errs

# sudo logs to local2 - keep a copy here too
local2.info                             /var/log/sudo.log

# Keep kernel messages local
kern.info                               /var/log/kern.log
```

[3] 更准确地说，它用“netloghost”作为它的主机别名。这样一来，几乎不需要重新配置就能对日志主机的标识进行调整。可以在**/etc/hosts** 中加入别名，或者在 DNS 中用 CNAME 记录建一个别名。参考 15.7.8 节了解有关 DNS CNAME 记录的更多信息。

www.TopSage.com

这样的配置并没有在本地保存太多日志消息。值得一提的是，如果 netloghost 已经关闭或者不可达，日志消息将会不可避免地丢失。用户可以在本地机器上保存某些重要消息的额外副本来应付这种可能发生的情况。

在安装了许多本地软件的站点中，许多消息可能会被不恰当地记录到 user 设备（facility）的 emerg 级别里。在本例中，第一行用 `user.none` 子句专门把 user/emerg 排除在外。

第 2 和第 3 行把所有重要消息转发到中央日志记录主机，从打印系统和全校刷卡访问系统（local1）来的信息都被排除在外。第 4 行把本地日志消息也传送到 netloghost。最后 3 项将保存打印机错误、**sudo** 日志消息和内核消息的本地备份。有关 **sudo** 的更多信息，请参见 3.4.2 节。

三、中央日志记录主机

本例针对 netloghost，它是由大约 7000 台主机构成的一个中等规模网络上的中央安全日志主机。

```
# syslog.conf file for master logging host

# Emergencies to the console and log file, with timing marks
*.emerg                          /dev/console
*.err;kern,mark.debug;auth.notice  /dev/console
*.err;kern,mark.debug;user.none    /var/log/console.log
auth.notice                      /var/log/console.log

# Send non-emergency messages to the usual log files
*.err;user.none;kern.debug       /var/log/messages
daemon,auth.notice;mail.crit     /var/log/messages
lpr.debug                        /var/log/lpd-errs
mail.debug                       /var/log/mail.log

# Local authorization stuff like sudo and npasswd
local2.debug                     /var/log/sudo.log
local2.alert                     /var/log/sudo-errs.log
auth.info                        /var/log/auth.log

# Other local stuff

local4.notice                    /var/log/da.log
local7.debug                     /var/log/tcp.log

# User stuff (the default if no facility is specified)
user.info                        /var/log/user.log
```

从本地程序和网络上的 **syslogd** 来的日志记录数据被写入到文件。在某些情况下，每种设备（facility）的输出结果都保存在它自己的文件中。

中央日志记录主机在写入每条消息时都为其生成一个时间戳。时间戳并不反映原来主机上的时间。如果用户拥有位于不同时区的主机，或者系统时钟不同步，那么时间戳就会有点儿误导性。

10.4.6 syslog 输出的样本

下面的日志文件片段取自科罗拉多大学计算机系主控 syslog 主机中的日志文件之一。

```
Dec 18 15:12:42 av18.cs.colorado.edu sbatchd[495]: sbatchd/main: ls_info()
    failed: LIM is down; try later; trying ...
Dec 18 15:14:28 proxy-1.cs.colorado.edu pop-proxy[27283]: Connection from
    128.138.198.84
Dec 18 15:14:30 mroe.cs.colorado.edu pingem[271]: maltese-
    office.cs.colorado.edu has not answered 42 times
```

www.TopSage.com

```
Dec 18 15:15:05 schwarz.cs.colorado.edu vmunix: Multiple softerrors: Seen 100
    Corrected Softerrors from SIMM J0201
Dec 18 15:15:16 coyote.cs.colorado.edu PAM_unix[17405]: (sshd) session closed
    for user trent
Dec 18 15:15:48 proxy-1.cs.colorado.edu pop-proxy[27285]: Connection from
    12.2.209.183
Dec 18 15:15:50 av18.cs.colorado.edu last message repeated 100 times
```

这个例子包含的记录项源自不同的主机（av18、proxy-1、mroe 和 coyote）和不同的程序：**sbatchd**、**pop-proxy**、**pingem** 以及 PAM（Pluggable Authentication Module，可插入式身份验证模块）库。

请注意节录的最后一行，它抱怨说有一条消息重复了 100 遍。为了使日志更加简短，syslog 通常会合并重复的消息，并用这样的总结来替换它们。但是，取得这个日志示例的主机还可以接受来自许多其他主机的日志记录，因此这条特别的消息就有一点儿误导性。在这一混合的日志中，这条消息实际上指的是前面来自 av18 的日志记录，而不是紧接在它前面的那条日志记录。

定期细读日志文件是个不错的想法。确切知道什么是正常的，以便在发生异常时可以很容易地发现它。还有一个更好的方法，安装一个日志的事后处理程序（例如 **swatch**）来自动捕获异常情况。请参阅 10.5 节。

10.4.7 使用 syslog 的软件

表 10.7 中列出了使用 syslog 的一些程序，还包括了这些程序的日志记录的设备（facility）和重要程度的级别，以及对每个程序的简短描述。有些行加了阴影是为了便于阅读。

表 10.7 使用 syslog 的软件

程　序	设　备	级　别	描　述
cron	cron，daemon	info	系统任务调用守护进程
cups	lpr	info-err	常用的 UNIX 打印系统
ftpd	ftp	debug-crit	FTP 守护进程（**wu-ftpd**）
inetd	daemon	err，warning	Internet 超级守护进程（Debian）
imapd	mail	info-alert	IMAP 邮件服务器
login	authpriv	info-err	登录程序
lpd	lpr	info-err	单行打印机守护进程
named	daemon	info-err	名字服务器（DNS）
ntpd	daemon，user	info-crit	网络时间守护进程
passwd	auth	notice，warning	口令设置程序
popper	local0	debug，notice	POP3 邮件服务器
sendmail	mail	debug-alert	邮件传输系统
ssh	auth	info	安全 shell（远程登录）
su	auth	notice，crit	切换 UID
sudo	local2	notice，alert	受限的 **su** 程序
syslogd	syslog，mark	info-err	内部错误、时间戳
tcpd	local7	debug-err	**inetd** 的 TCP 封装程序
vmlinuz	kern	所有级别	内核
xinetd	可配置	info (默认)	**inetd** 的变体（Red Hat、SUSE）

有了所有这些信息，用户应该非常清楚哪些消息应该保留、哪些应该删除了，不是吗？但是，可

www.TopSage.com

能未必如此。实际上，用户必须知道什么样的日志级别对自己的系统有用。最好的方式是首先让日志记录超过需要，然后去掉那些不想要的内容。等到产生数据的平均速率令您满意时就可以了。

10.4.8 调试 syslog

logger 命令用于从 shell 脚本提交日志项。用户也可以用它来检测 **syslogd** 的配置文件的变化。例如，如果已经添加了一行：

```
local5.warning        /tmp/evi.log
```

而且想看看它是否起了作用，则运行：

```
$ logger -p local5.warning "test message"
```

含有“test message”的一行记录项应该写到**/tmp/evi.log** 文件中。如果没有写入，那么可能是忘记了给 **syslogd** 发送挂起信号。

把日志记录发给控制台设备**/dev/console**、任何虚拟终端或支持流控的端口的时候应该小心。如果有人在控制台键入了<Control-S>，那么控制台的输出将会停止。所有对 **syslog** 的调用都将受阻，系统将变得非常缓慢。检查这种性能下降情况有一个好方法，即通过 **logger** 向控制台发送一条 syslog 消息。如果 **logger** 挂起了，则用户需要找到出了问题的端口，键入<Control-Q>，然后重新考虑日志记录的策略。

在控制台上做日志记录的另外一个缺点是，由一个大问题产生的大量消息可以使控制台变得不稳定，而此时正是最需要它的时候。如果没有优化过控制台驱动程序，又采用某种帧缓冲类型的话，大量的消息确实会让整个系统无法使用。

根据控制台的设置和管理情况不同（例如，通过一台控制台服务器），控制台的日志记录也应该有一些安全措施。

10.4.9 在程序中使用 syslog

库例程 **openlog**、**syslog** 和 **closelog** 能够让程序使用 syslog 系统。这些库例程有 C、Perl、Python 和 PHP 的版本，这里只讨论 Perl 的接口。

在 Perl 脚本的开始处加上下面的一行，就可以引入库例程的定义：

```
use Sys::Syslog;
```

openlog 例程使用指定的设备（facility）名称初始化日志记录：

```
openlog(ident, logopt, facility);
```

消息按照 *logopt* 指定的选项被记录下来，并以标识字符串 *ident* 开头。如果没有用过 **openlog**，那么默认 *ident* 为当前用户名，*logopt* 为一个空字符串，而 *facility* 为“user”。*logopt* 字符串应该包含一个由逗号隔开的选项清单，这些选项在表 10.8 中列出。

表 10.8　　例程的日志记录选项

选　项	含　义
pid	在每条日志消息上包含当前进程的 PID
ndelay	立即连接到 **syslogd**（不用等到要发消息的时候再连接）
cons	如果 **syslogd** 不可达，就把消息发送到系统控制台
nowait	不用等待（**wait(3)**）派生子进程来写控制台消息

www.TopSage.com

例如，合理的 **openlog** 调用可以是：

```
openlog("adminscript", "pid,cons", "local4");
```

syslog 例程发送一个消息至 syslogd，将其以指定的优先级记录：

```
syslog(priority, message, ...);
```

来自 **openlog** 调用的日期、时间、主机名称和 *ident* 字符串都预先加入到日志文件的消息中。*message* 后面可跟各种其他参数，以形成一个 **printf** 风格的输出格式，包括文本和其他变量的列表，例如：

```
syslog("info","Delivery to'%s'failed after %d attempts.", $user, $nAttempts);
```

特殊符号%m 把 **errno** 的当前值（最新的错误代码）扩展为一条错误消息。

形如 "*level* | *facility*" 的优先级字符串既设置了严重性级别，也设置了设备名称。如果没有调用 **openlog** 并指定一个 *ident* 字符串，**syslog** 例程还会检查 *message* 是否具有一条标准 UNIX 错误消息的形式，例如：

```
adminscript: User "nobody" not found in /etc/passwd file.
```

如果符合这个格式，冒号之前的那一部分就被秘密地采用为 *ident* 字符串。这些有用（但没有在文档中正式说明）的特性使得调用 **openlog** 并不是必不可少的了，但调用 **openlog** 仍然是个不错的想法。在一个地方（调用 **openlog** 时）指定设备的名称，比在整个代码中不断重复设备名称要好。

closelog 例程将关闭日志记录通道：

```
closelog();
```

如果想以不同的选项重新打开日志记录通道，就必须调用这个例程。在程序退出时调用 **closelog** 是一个很好的形式，但是这样做并非必不可少。

下面有一个完整的例子：

```
use Sys::Syslog;

openlog("adminscript", "cons,pid", "user");
syslog("warning","Those whom the gods would destroy, they first teach Basic.");
closelog();
```

这一小段脚本将产生下面的日志记录（191 是 **admincript** 的 PID）：

```
Dec 28 22:56:24 moet.colorado.edu adminscript[191]: Those whom the gods
    would destroy, they first teach Basic.
```

10.5 分析日志文件

syslog 对于日志消息的排序和路由选择非常有用，但当所有一切都安排妥当之后，最终的结果仍然是一堆日志文件。虽然这些文件可能包含了所有有用的信息，但在出现问题时，这些文件并不会自己找上门来。需要另一层软件来分析日志并保证不会在众多日志中找不到重要的信息。

可以使用各种免费工具来弥补这个漏洞，它们中的大多数都很相似：它们可以扫描最近的日志项、把它们与一个正则表达式的数据库进行匹配，并用某种引人注目的方式处理重要的信息。有些工具可以以电子邮件形式发送报告，其他一些工具可以配置成发出声响、用不同的颜色打印日志项或者向呼机发送信息。这些工具的差别主要在于它们的灵活性程度和现成的匹配模式数据库的大小。

www.TopSage.com

有两种比较常用的日志事后处理程序，一个是 Todd Atkins 编写的 **swatch**，另一个是 Craig Rowland 编写的 **logcheck**。这两个程序都能从 sourceforge.net 找到（**logcheck** 随 sentrytools 软件包发布：sourceforge.net/projects/sentrytools）。

swatch 是一个 Perl 脚本，它从一个配置文件获得它的执行顺序。它的配置语法相当灵活，并且还能使用 Perl 的所有模式匹配功能。虽然 **swatch** 可以一次处理一个完整的文件，但主要还是让它自己保持运行，以便在新消息到达时可以用 **tail -f** 及时查看。**swatch** 的一个不足之处是用户必须从头建立自己的配置文件，它不知道某些特殊系统，也不理解这些特殊系统可能生成的日志消息。

logcheck 是用 **sh** 写出的一个更基本的脚本，与它一起发布的还有一个 C 程序，**logcheck** 用它帮助自己在一个日志文件中记录自己的位置。由于 **logcheck** 知道它从日志文件中读取了多少内容，所以降低了启动或者关机时丢失消息的可能性。此外，**logcheck** 可以在 **cron** 中以一定间隔运行，而不是连续运行。

logcheck 还带有用于几种不同 UNIX 和 Linux 版本的数据库样例。即使用户不想实际使用这些脚本，参考它们的格式看看能否从中借鉴一些以供自己使用也是值得的。

这两个工具都有一个弊病，即一次只能用于一个日志文件。如果用户的 syslog 配置将消息分类到多个不同的文件中，则可能要把某些消息复制到一个频繁被截短的中央文件里，然后把汇总文件送给事后处理脚本来处理。这比建立一个复杂的脚本网络来处理多个文件要容易一些。

另一种值得一提的工具是 Kirk Bauer 写的 **logwatch**。与其说它是一个即时监测和报警工具，还不如说是一个日志总结工具。但它的优点有：相当简单，Fedora 和 Red Hat 默认都安装它，并且有 RPM 包和 APT 软件包可用。

SEC（简单事件关联程序，Simple Event Correlator）是一种不同类型的日志管理工具。它是一个 Perl 脚本，能够从文件、有名管道或者标准输入读取数据行，然后用正则表达式匹配它们，把它们转换为各类“输入事件”。配置规则接着规定了如何把输入事件变为输出事件，比如执行了某个脚本、向某个指定的管道或者文件发了一则消息。

从 kodu.neti.ee/~risto/sec 可以获得 SEC 的软件发布，它带有一份内容丰富的手册页，其中还有例子。在这个网站上还有更多别的例子。SEC 并不像前面所列的那些工具一样是现成可用的，但它是一个用来构建定制的日志分析工具的好基础。

不管用户使用什么系统来扫描日志文件，下面这些情况都应该留心，并应该立即引起管理员的注意。

- 大多数与安全相关的消息都应该及时检查。对于监视登录、调用 su 和 sudo 的失败企图，以便在潜在的入侵发生之前先发现它们来说，这样做通常会有帮助。如果只是某人忘记了他的口令（这是经常发生的事情），那么及时而主动地提供帮助将给别人留下很好的印象，并且能巩固您具有敏锐洞察力的声望。
- 磁盘已满的消息应该突出，并立即采取措施。充满的磁盘常常会使所有工作陷于停顿。
- 不管怎样，多遍重复的消息都应该特别注意。

10.6 习题

E10.1 保留老日志文件的主要理由有哪些？

E10.2 **lastlog** 和 **wtmp** 之间的区别是什么？每种日志合理的轮换策略是什么？

E10.3 剖析和理解下面的 **syslog.conf** 文件：

```
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
```

www.TopSage.com

它看上去合理吗？

E10.4 看一遍您的日志文件，注意 **named**（DNS 名字服务器）的日志记录项。有什么机器试图动态更新您的名字域吗？它们取得成功了吗？（可能需要 root 权限）

★E10.5 您可以从哪个地方找到您机器上的引导日志？影响系统引导时日志记录机制的因素有哪些？**klogd** 是如何解决这些问题的？

★E10.6 研究您的站点所采用的日志策略，包括日志文件轮换策略。为日志记录投入了多少磁盘空间？日志文件要保留多久？您可以预见到站点策略显出不足的情形吗？您建议的解决方案是什么？（需要 root 权限）

★E10.7 有些日志消息极为重要，系统管理员应该立即查看它们。您会安装什么系统来尽可能快地做到这一点？

★E10.8 编写一个 C 程序或者 Perl 脚本，用“user”设备把消息提交给 syslog（可能需要 root 权限）。

www.TopSage.com

大家网
TopSage.com

第 11 章　软件和配置管理



虽然 Linux 发行版本正在迅速变得功能更加完善和界面更加友好，但是软件配置管理仍然是系统管理领域内的一项重要工作。除了安装现成的系统，还必须为满足本地环境的特殊需求而做任何调整。系统管理员一般必须要执行下面所有的任务：

- 自动批量安装操作系统；
- 完成系统的本地化；
- 给系统打补丁且保持系统的更新；
- 管理附加的软件包。

配置现成的发行版本或者软件包，以求符合自己的需要（以及本地安全规定、文件存放和网络拓扑的需要），这个过程往往称为“本地化”。本章研讨的一些技术和应用有助于减少安装软件的痛楚，让这些工作收放自如。

11.1　Linux 基本安装

目前的各种 Linux 发行版本都提供很直观的基本安装步骤。您可以从光盘或者软盘启动，回答几个问题，告诉安装程序要把哪些软件包拷贝到硬盘上。大多数 Linux 发行版本还有不错的安装指南。



企业版的 Red Hat 带有印刷好的安装手册。这本手册（以及其他 Red Hat 手册）也能在网上 http://www.redhat.com/docs/manuals/这个地址找到。



Fedora 有自己的建档项目，但是因为当前的 Fedora 发布还保持着和 Red Hat Enterprise Linux 密切的联系，所以 Fedora 的志愿者们似乎还不情愿花时间复制现有的 RHEL 文档。在寻找 Fedora 问题的答案时，Red Hat 手册通常是一个很好的起点。

www.TopSage.com



盒装销售的 SUSE Linux Enterprise 版也带有安装手册。这些文档可以从网上 www.novell.com/documentation/suse.html 这个地址获得。openSUSE 的类似文档可以从 www.openSUSE.org/Docuemtation 获得。



Debian 的安装手册在这个地址能找到：www.debian.org/releases/stable/installmanual。



Ubuntu 以 wiki（维基）形式很好地总结了的安装选项，地址为：http://wiki.ubuntu.com/Installation。

如果您必须要一次安装多台计算机，那么您会发现自己很快就达到了交互式安装的极限。首先，这样做效率不是太高。（您还能重复性地从安装时成百上千的软件包中确切地选出一组软件包吗？）对于这类问题，可以用一个本地化复查表把选错的机会减少到最小，但即便是这样的方法也不能避免一定选对。交互式的安装也费时费力。

我们举例的发行版本所采用的安装程序都是脚本，所以另一种做法是做一张软盘或者光盘，里面包含您想要的配置。这种办法解决了效率问题，但是它仍然不能很好地扩展。如果您要安装数百台机器，那么做数百张配置盘来用也是极其沉闷的工作。

通过网络自动安装也是可能的，对于有着数百乃至数千系统要升级的站点来说，这通常是最方便的方法。我们所举的所有发行版本都能够以这种方式安装，但是在平稳地进行自动安装之前，确实要对现成的系统做些配置。

还有几种开放源代码的开发项目致力于做出第三方的网络安装程序，SystemImager 就是一个例子。本章将详细讨论这些系统。

11.1.1 从网络引导 PC

从网络引导系统的做法可以不必从硬盘、软盘或者 CD-ROM 光盘，而是完全通过网络来引导一台计算机。这对于安装来说很方便，因为它意味着您可以坐在桌子旁边，不必走到机器那里插入 CD-ROM 光盘或者软盘，就可以从网络上引导位于别处的一台机器。

专门制造 UNIX 计算机的制造商长期以来一直都支持网络启动，但是直到最近，这项特性才成为 PC 上的标准。在研究定制型的引导 PROM 多年之后，Intel 公司已经制订出 PXE（Pre-boot eXecution Enironment）标准，以此给予用户很大的便利。虽然 PXE 还不完善，但是不好的标准（一般）比没有标准强。

PXE 的作用就像是在网卡上 ROM 里的一个小小操作系统。它通过一个标准 API 把自己的网络功能提供给系统 BIOS。有了 PXE 和 BIOS 两者的合作，就有可能在任何有 PXE 的 PC 上，让一个引导加载程序不需要给每个网卡提供专门的驱动程序，就能通过网络启动 Linux。

PXE 协议的对外（网络）部分很直观易懂，和其他体系结构上采用的网络引导过程类似。一台主机广播一则设置了 PXE 选项的 DHCP“发现”请求，然后一台 DHCP 服务器或者代理服务器用一个包含有 PXE 选项的 DHCP 包来响应它（引导服务器的名字和引导文件的名字）。客户机通过 TFTP 下载它的引导文件（或者也可以采用多播 TFTP），然后执行这个引导文件。参见 12.9 节了解有关 DHCP 的更多知识。

虽然 PXE 规范不能完全向后兼容普通的 DHCP，但在实际中，许多现代的 DHCP 服务器能够提供 PXE 引导服务。

不同的发行版本内部对 PXE 支持的程度也不相同。但这是一个发展活跃的领域，基于 PXE 功能的系统正在变得越来越普遍。介绍 PXE 的文档从很多地方都能得到。

www.TopSage.com

11.1.2 为 Linux 设置 PXE

现在有几种基于 PXE 的网络引导系统，但是目前效果最好的是 H. Peter Anvin 的 PXELINUX，这是他的通用引导加载程序 SYSLINUX 的一部分。从 syslinux.zytor.com 可以得到这个软件。

PXELINUX 提供了一个引导文件，用户把这个文件安装到服务器的 **tftpboot** 目录下，当 PXE 起作用的时候，就会被下载到正在启动的 PC 上。这台 PC 接着执行这个引导文件，从服务器下载它的配置。下载的配置规定了该使用哪个内核。这一系列的步骤不用外部干涉就能完成，如果愿意，用户也可以建立一个自己定制的引导菜单。

PXELINUX 使用 PXE API 下载，因此在引导过程中始终不受硬件影响。它也不仅限于引导 Linux——它还能引导其他操作系统，如果使用 MEMDISK 内核，甚至还能引导软盘映像。MEMDISK 也是 SYSLINUX 软件包的一部分。

在服务器端一定要使用 ISC（Internet Systems Consortium）的 DHCP 服务器。如果使用了不一样的服务器，那就必须配置额外的 DHCP 选项，或者使用服务器的配置文件里的数字表示。



Red Hat Enterprise Linux 3 System Administration Guide 的第 14 章内容全都讲的是 PXE 和网络引导。参见 http://www.redhat.com/docs/manuals。



运行命令 **system-config-netboot**，RHEL 对应的免费版本 Fedora 就能让用户访问图形界面的网络引导配置系统。



SUSE 没有带任何 PXE 软件包，但您可以借用 Red Hat 的 PXE 软件包。可以很容易让 YaST 配合 PXE 使用，虽然 SUSE 并未正式支持这种配置。



Debian 的安装程序也可以毫无问题地从网络引导，只要从 www.debian.org/ devel/ debian-installer 下载网络引导所用的内核映像即可。



在编写本书英文版的时候，Ubuntu 6.06 版的正式安装手册还没有完成。不过，您可以在 doc.ubuntu.com 找到一份草稿。指导安装该发行版本的 wiki 网页地址为：https:// wiki.ubuntu.com/Installation/Netboot。请参考其中的 Installation/Netboot 和 Installation/ LocalNet 两个专题的内容。

11.1.3 非 PC 的网络引导

PXE 是 Intel 公司的一个产品，它只限于在 IA-32 和 IA-64 的硬件上使用。其他体系结构有它们自己的网络引导方法（几乎都比 PXE 更完善）。对于网络引导来说，Linux 的范围已经超出了 Intel 的体系结构，许多“专用于”UNIX 的网络引导系统现在都可以选择从网络引导 Linux，而不是它们原本适用的操作系统，情况的发展出现了有趣的颠倒现象。

讨论各种体系结构之间的诸多不同的确超出了本书的范畴，但是网络上的各种资源能够提供帮助。

SPARC 机器和大多数 PowerPC 计算机都使用 Open Firmware（开放性的固件），用它从网络启动很容易（键入 **boot net**）。UltraLinux FAQ（www.ultralinux.org）有一份介绍如何在 SPARC 处理器上通过网络启动 Linux 的指南。Alpha 机器一般使用 SRM 控制台软件来引导 Linux，*SRM HOWTO* 介绍了如何使用这些功能。对于 Mac、RS/6000 和其他基于 PowerPC 的机器来说，网络引导的过程则取决于硬件和所用的引导加载程序，所以要查看引导加载程序的文档。AMD64 机器和它们的“兄弟”——x86 PC 一样，也使用 PXE。

www.TopSage.com

11.1.4 Kickstart：RHEL 和 Fedora 的自动安装程序

Red Hat 用于执行自动安装的工具叫做 Kickstart。Kickstart 是 Red Hat 标准安装程序 Anaconda 的脚本化接口，它随 Red Hat 发行版本和 RPM 软件包的不同而不同。Kickstart 同 SystemImager 以及其他原始映像（raw image）复制程序不一样，它不但灵活，而且相当智能，能自动检测到系统的硬件。

如果您需要做大量的本地化工作，可以把需要做的任何改动作为安装后执行脚本的一部分。虽然这种做法可行，但安装后执行的脚本会变得又大又不易维护。另一种做法是做一个 RPM 软件包，其中包含您的本地定制信息。我们建议采用 RPM 的路子，它可以方便版本的升级，易于日后的更新，而且提供了依赖性支持以及软件打包系统具有的其他好处。参考 11.3 节了解有关软件打包的更多信息。

一、建立 Kickstart 配置文件

Kickstart 的行为是由一个配置文件来控制的，它一般叫做 **ks.cfg**。这个文件的格式直观易懂。如果您偏爱可视化，那么 Red Hat Enterprise Linux 带有一个方便的 GUI 工具，叫做 **redhat-config-kickstart**，在 Fedoar 上叫 **system-config-kickstart**，它能让您通过点击的方式配置 **ks.cfg**[1]。

ks.cfg 文件也相当容易用程序来生成。例如，假定您想要在服务器和客户机上安装不同的两组软件包，而且您也需要处理为多个办事处定制的稍有不同的安装。您就可以编写一个 Perl 的小脚本，它使用一组主控参数为服务器和每个办事处的客户机生成一个配置文件。改动补足的软件包就变成了一件改变这个 Perl 脚本的事情，而不是改变每个 config 文件的事情。甚至还有一些情况需要您为每台主机都生成一个专门的配置文件。在这种情况下，您肯定想自动生成这些文件。

The Red Hat Enterprise Linux System Administration Guide 的第 2 部分里有一份很好的 Kickstart（包括所有可用选项的一份清单）手册。它可以从下面的地址获得：http://www.redhat.com/docs/manuals。

关于 Kickstart 的配置文件要注意一点：如果您犯了一个错误，莫名其妙的 Python 对此所响应的诊断信息里可能包含也可能不包含可以指出错误之处的一则隐藏消息。使用 **redhat-config-kickstart** 或者 **system-config-kickstart** 的主要优点之一就是它肯定能生成一个至少在语法上是有效的配置文件。

Kickstart 配置文件依次由 3 部分组成。第一部分是命令段，它指定诸如语言、键盘和时区这样的选项。这一部分也能用 url 选项指定发行版本的来源（在下面的例子里，来源是叫做 installserver 的主机）。

下面是一个命令段的完整例子：

```
text
lang en_US                      # lang is used during the installation...
langsupport en_US               # ...and langsupport at run time.
keyboard us                     # Use an American keyboard.
timezone --utc America/EST      # --utc means hardware clock is on UTC (GMT)
mouse
rootpw whatever
reboot                          # Reboot after installation. Always a good idea.
bootloader --location=mbr       # Install default boot loader in the MBR.
install                         # Install a new system instead of upgrading.
url --url http://installserver/redhat
clearpart --all --initlabel     # Clear all existing partitions
part / --fstype ext3 --size 4096
part swap --size 1024
part /var --fstype ext3 -size 1 --grow
network --bootproto dhcp
auth --useshadow --enablemd5
firewall --disabled
xconfig --defaultdesktop=GNOME --startxonboot --resolution 1280x1024 --depth 24
```

[1] 注意，Fedora 把所有名为 redhat-config-*的工具都改为 system-config-*。名字上的变化是扩大 Fedora 和 Red Hat 间距离的一项努力，但是名字的变化（还）没有反映出下面的技术不同。

www.TopSage.com

Kickstart 默认使用图形方式，这种方式不能用于无人照看的安装。在这个例子开头的关键字 text 解决了这个问题。

rootpw 选项设置新机器的 root 口令。默认以明文指定这个口令，这会带来安全问题。您可以使用--iscrypted 标志来指定一个加密过的口令，不过，这种方式不支持 MD5 口令。

clearpart 和 part 指令指定了一系列分区以及它们的大小。您还可以指定其中一个分区能够扩大填满磁盘上剩下的全部空间。这个功能能够很方便地接受硬盘容量不同的系统。

第二部分列出了要安装的软件包，这一段的开头是一条%packages 指令。软件包的清单可以包含单个软件包、像@ GNOME 这样的软件集合、或者是表示全部安装的记号@ Everything。在选择单个软件包的时候，只需指出软件包的名字，不必带版本或者.rpm 扩展名。下面是一个例子：

```
%packages
@ Networked Workstation
@ X Window System
@ GNOME
mylocalpackage
```

在 Kickstart 配置文件的第三部分中，您可以指定要 Kickstart 执行的任何 shell 命令。可能有两组命令：用%pre 引入在安装之前运行的命令，用%post 引入在安装之后运行的命令。这两部分都会要求系统能够解析主机名，所以如果您想要访问网络，那么最稳妥的做法就是使用 IP 地址。此外，安装后运行命令是在一个 **chroot** 过的环境里运行的，所以它们都不能访问安装介质。

二、建立一个 Kickstart 服务器

Kickstart 希望其文件的布局和它们在发行光盘上的布局一样。它要求它的软件包都保存在服务器上称为 **RedHat/RPMS** 或者 **Fedora/RPMS** 的目录下面。您可以随意向这个目录加入您自己的软件包。不过，这里要注意两个问题。

首先，如果您告诉 Kickstart 安装所有的软件包（在 **ks.cfg** 文件里的%packages 部分里用了一条@ Everything 指令），它就在已经装完基础软件包之后按字母顺序安装附加的软件包。如果软件包所依赖的其他软件包没有在基础包之中，那么可能要把您的软件包叫做 **zzmypackage.rpm** 这样的名字，以确保它在最后安装。

如果您不想安装全部的软件包，要么可以在 **ks.cfg** 文件中的%packages 部分列出要补充的各个软件包，要么可以加上您自己的一个或者多个软件包集合。集合的清单由@ GNOME 这样的指令项来指定，它代表一组预先定义好的软件包，其中的成员一一列举在服务器上的文件 **RedHat/base/comps** 或者 **Fedora/base/comps** 里。遗憾的是，**comps** 文件的格式没有好的文档说明。软件包集合是以 0 或者 1 开头的行（数字表明这个集合是否是默认选中的）。一般而言，乱改标准的软件集合可不好。我们建议您不要动它们，保持 Red Hat 定义它们的原样，而在 **ks.cfg** 文件里明确地命名追加的所有软件包。

三、把 Kickstart 指向配置文件

一旦已经建好了一个配置文件，可以有两种方法来让 Kickstart 使用它。公认的方法是用一张引导软盘或者光盘来引导，通过在开始的 boot:提示处输入 **linux ks** 来要求进行 Kickstart 安装。如果您没有提供任何其他参数，那么系统就使用 DHCP 来确定它的网络地址。接着它就会知道 DHCP 引导服务器，得到引导文件选项，尝试以 NFS 安装引导服务器，使用引导文件选项的值作为它的 Kickstart 配置文件。如果没有指定引导文件，系统就查找一个称为**/kickstart/***hostipaddress*-**kickstart** 的文件。

另一种方法是，可以通过提供一条路径作为 **ks** 选项的参数，以其他方式通知 Kickstart 得到它的配置文件。这会有几种可能性。指令

```
boot: linux ks=http:server:/path
```

告诉 Kickstart 使用 HTTP 而不是 NFS 下载文件。使用 **ks=floppy** 告诉 Kickstart 在本地软盘驱动器里查找 **ks.cfg** 文件。

www.TopSage.com

要完全不用引导介质，您需要进而使用 PXE。参考 11.1.2 节了解有关它的更多信息。

TopSage.com

11.1.5 AutoYaST：SUSE 的自动安装工具



YaST2 是 SUSE 全能的安装和配置工具。它有一个很不错的 GUI，在安装单个系统的时候很好用。较早的 SUSE 发行版本可以通过普通的 YaST 自动安装，但是结果并不能让所有人满意。SUSE 8 引入了一个自动安装 SUSE 的工具，它叫做 AutoYaST。在本书介绍的所有发行版本之中，它是功能最强的系统自动安装软件。从 www.SUSE.com/~ug 能够获得它的详细文档。

SUSE 把自动安装过程分为 3 个阶段：准备、安装和配置。一开始的准备阶段是用 AutoYaST 模块来执行：

```
$ /sbin/yast2 autoyast
```

这个模块帮助您指定您所要设置的细节。运行它的结果是产生了一个 XML 控制文件，这个文件告诉安装程序如何配置一个 SUSE 系统。在之前提到的网上文档里介绍了该文件的结构。

有两条捷径能加快配置过程的速度。AutoYaST 模块可以读入 Red Hat 的 Kickstart 配置文件，帮助用户从“遗留的”老系统进行升级。如果您想要复制目前正在工作的机器的配置，那么有一个选项也能自动完成这项工作。

实际安装需要 3 项网络服务：

- 在要安装的机器的同网段内有一台 DHCP 服务器；
- 有一台 SUSE 安装服务器或者软件库；
- 有一台为安装提供配置信息的服务器。

这 3 台服务器中的最后一台能够根据选择，以 HTTP、NFS 或者 TFTP 提供配置文件。

在大多数基本安装里，您要为将要安装的每台机器做一个控制文件。AutoYaST 使用客户机的 IP 地址来判断要使用哪个配置文件。如果您必须要安装一系列有轻微不同的系统，那么这种方法效率不高。

您可以通过使用一个规则系统，创建更为复杂的设置。根据系统的属性，如硬盘大小、主机号，或者是否有 PCMCIA，针对目标系统匹配若干不同的控制文件。所有选出的文件要合并到一起，在出现冲突的时候，后面的控制文件则覆盖前面文件的内容。（一个控制文件不必非要规定一个系统的配置的所有方面，所以这样的合并才有意义。）

控制文件也能根据主机名或者 IP 地址范围规定机器的“分类”，而每一分类可能还有与之相关的另一从属控制文件。机器可以不属于任何分类，也可以属于一个或者多个分类，而它们的配置则由所有相关的分类控制文件合并而成。

幸好 AutoYaST 能够把多个控制文件的内容合并到一起，它的结构能做到以最小的冗余定义复制的设置。XML 控制文件读起来有点麻烦，但是由任何常用的 XML 处理工具来处理和编辑这些文件则很简单。

11.1.6 Debian 和 Ubuntu 的安装程序



Debian 的安装程序（名字叫的也挺贴切，debian-installer）完全是为这个版本重新写的。现在它用起来是个相当有意思的工具。虽然 Ubuntu（我们认为是明智地）保留了所有 Debian 化的命令名，但它也用这个安装程序。

www.TopSage.com

这个新的安装系统采用一种模块化的设计，用 C 和 shell 脚本配合编写。用两张软盘就能在计算机上运行它，它只占 32MB 的内存，可以直接从网上把安装所需的所有文件（包括它需要的文件的清单）都下载下来。

Debian 安装程序的所有交互性步骤都采用 **cdebconf** 这个工具来决定向用户提出哪些问题，使用什么样的默认答案。给 **cdebconf** 提供一个预先确定的答案库，就能让安装程序完全自动执行。用户可以手工（数据库只是一个文本文件）创建这个数据库，也可以在我们的一个示例系统上做一次交互式的安装，然后用下面的命令把您给 **cdebconf** 的答案都转存下来：

```
# debconf-get-selections --installer > config.cfg
# debconf-get-selections >> config.cfg
```

让配置文件从网络上能够下载，然后在安装系统的时候用下面的内核参数把它传给内核：

```
preseed/url=http://host/path/to/preseed
```

如果想要自动部署 Debian，也要看一下 **cdebootstrap** 这个软件包。它和 Debian 安装程序以相同的代码为基础，能让用户在本地硬盘的子目录里创建一个安装映像，从 Debian 的资源库中直接下载所需的软件包。这个功能在用户自己设计的自动配置脚本中用起来很方便。

在 Debian 早先的发布里，安装程序是被广为诟病的地方。许多系统管理员必然会去实现自己的安装和配置系统，下面的地址列出了各种这样的实现：

www.linuxmafia.com/faq/Debian/installers.html。

另一种标准的 Debian 安装工具是一个称为 FAI（代表全自动安装的意思）的软件包。它的主页是 www.informatik.uni-koeln.de/fai。这些替代系统现在都不如以前那么需要了，但是如果决定与标准的安装程序分道扬镳，那么它们仍然值得看看。



虽然 Ubuntu 是 Red Hat 家族之外的一种发行版本，但是它却兼容 Kickstart 控制文件，把这些文件嫁接到自己的下层安装程序上。为了创建这些文件，它还带有 **system-config-kickstart** 这个工具。Ubuntu 上的 Kickstart 功能还没有达到 100%可靠，但是看上去完全够用了。对 Kickstart 的兼容性还不能和 Debian 共享。

11.1.7 从主控系统安装

如果不得使用发行版本的标准安装程序，那么可以考虑站点的硬件一致性是否够好，好到只要把操作系统从一个机器复制到另一个机器即可。复制要比传统的安装快得多，但它对一致性的要求稍微强了一些。

您可能会发现下面大致列出的几条会有帮助：

- 用发行版本的正常安装步骤或者像 Debian 的 **debootstrap** 这样的脚本，安装和配置主控系统；
- 用一张启动光盘（比如 Knoppix2[2]）或者采用基于 PXE 的方法启动机器，要把操作系统复制到这台计算机上；
- 交互式地使用 **cfdisk** 命令或者从脚本调用 **sfdisk** 对硬盘分区；
- 创建文件系统、初始化交换空间，并且在**/mnt** 下安装文件系统；
- 用 **ssh** 和 **rsync**（参见 17.3.2 节）命令把主控安装的操作系统复制到新分区的硬盘上；
- 编辑所有要定制的文件（例如，**/etc/mdules** 或者**/etc/fstab**），抑或更好一些，自动执行这个

[2] Knoppix 是一个 Debian 的版本，这个版本直接从可启光盘（CD/DVD-ROM）运行，不需要安装到硬盘上。它在安装或者营救系统的过程中，或者在处理安全事件的时候使用很理想。许多别的发行版本（例如，Ubuntu）也开始能从光盘上运行了。

www.TopSage.com

过程；

- 在硬盘的 MBR（主引导记录）里安装 GRUB 或者 LILO，让新机器能够启动，参见 2.3 节的内容了解详情。

11.2 无盘客户机

在 20 世纪 80 年代，当时的硬盘还很贵，许多站点部署了没有本地硬盘的工作站。这些计算机称为“无盘客户机”，它们要从一台网络服务器安装自己的文件系统。现如今，硬盘非常便宜，几乎没人还在用无盘客户机了。然而它的整个思想就和 9 道磁带或者点阵打印机一样显得很精巧。

不过且慢，无盘运行仍然可以存在，对于某些安装类型来说，它实际上是一种合理的选择，其优点如下。

- 在无盘环境下，所有的数据都保存在服务器。无盘客户机绝不会因为它们的配置信息访问不到就不运行了，所以系统管理步骤里不需要把不运行的机器作为一种可能发生的特殊情况考虑进来。
- 真正的本地配置信息量保持在最少的程度。几乎没有什么管理操作要求有人亲自在客户机前面执行。
- 新客户机在它们运行甚至出现之前，就可以在服务器上先配置。客户机的行为就好像是个逻辑设备而不是物理设备。
- 对客户机的配置所做的重大改变可以放在需要重启的时候执行。更新过的根目录结构可以放在正在使用的目录结构旁边，等到合适的机会投入使用。使用的软件和更新系统的概念更简单，同时又提供了更快更可靠的服务。
- 对于许多应用来说，把能买得起的昂贵且复杂的 RAID 盘阵加到服务器上，这比起廉价的本地硬盘，即便把网络延迟考虑在内，速度也还是会更快。网络服务器也可能变得更可靠。硬盘会出故障，但是冗余服务器的设置却能保护好整个无盘客户机的网络。
- 没有哪条规则要求无盘客户机实际中一定不能有硬盘。如果愿意，可以用本地硬盘做交换空间或者高速缓存。在现代概念里，不是说客户机不能有物理硬盘，而是说它们不用这些硬盘保存需要系统管理的东西。
- 许多应用都是受限于 CPU 的应用，它们不特别依赖硬盘速度。其他有些应用有可能格外依赖硬盘，但增加内存后也可以把它们转为受限于 CPU 的应用。Web 服务器、数据库服务器和网络协议服务器都有可能落入到这类应用当中。

在无盘环境下，客户机的根文件系统一般较小，且供自己专用。多台客户机可以共享一个共同的 **/usr** 目录，因为这个目录装的信息与具体计算机无关，在日常操作中从来不用向这个目录写数据。

Red Hat Enterprise Linux 是唯一一个带有一个标准化的系统，能为该服务器之外配置无盘客户机的 Linux 发行版本。但即便没有得到厂商的很好支持，配置无盘客户机也不像造火箭，自己动手并不难。许多系统都有一种通过网络启动 X 窗口终端的机制，可以以此为起点来启动工作站。

如果发行版本没有提供设置无盘客户机的说明，那么一开始可以把一个新装机器的根文件系统复制到您的服务器上，以此为基础来克隆新的机器。研究明白如何通过网络用一个 NFS 的根目录引导系统。在所有的客户机之间共享计算机上文件系统的只读部分（例如，**/usr**）。

克隆一个客户机的根目录时，可能还要做下面的定制工作：

- 文件系统的安装表；
- 硬件设备的支持（如果网络并不是同构网络的话）；
- 网络配置（可能需要）；

www.TopSage.com

- 启动脚本（可能需要）。

普通计算机经常带有廉价的 IDE 硬盘，可以把这些硬盘用于交换空间和临时用的分区。

如今便宜的 RAM 给无盘节点带来了很大的实惠，可以考虑用一个基于 RAM 的文件系统给**/tmp**和**/dev** 目录做备份（或者作为频繁使用文件的高速缓存，在启动时预先载入内存）[3]。把**/dev** 这个目录树放在 RAM 里，就能大大提高性能，因为喜欢访问**/dev** 目录树的程序不会因为 NFS 的延迟而影响性能。参考 25.3 节了解更多有关 RAM 盘的知识。

11.3 软件包管理

Linux 的发行版本都采用了某种形式的软件包系统来简化配置管理工作。软件包传统上一直用来发布软件，但它也可以给配置文件和管理数据打包。它们比起传统结构的.tar.gz 存档文件有几个优势。其中最重要的可能要算它们能让安装过程尽可能成为不可分割的原子操作。如果发生错误，可以卸载软件包，或者重新安装软件包。

UNIX 厂商经常以特殊的方式来把补丁发布给他们的客户，但是 Linux 发行商则利用了他们的标准软件包管理机制。发行商只需发布一个更新后的软件包，就等于发了一个补丁。客户安装新版软件包的同时就把老版替换掉了。

软件包的安装程序一般都知道配置文件，正常情况下不会覆盖由系统管理员所做的本地定制信息。它们要么会备份它们改动过的配置文件，要么以不同的名字提供配置文件的样例（例如，**pkg.conf.rpmnew**）。如果您发现新安装的软件包破坏了系统上的什么地方，至少从理论上讲，您可以卸载它，恢复系统原本的状态。当然理论不等于实际，所以没有先测试的话，不要在一个生产系统上尝试安装新软件包。

软件打包系统规定了一种依赖模型，要求软件包的维护程序保证其应用软件所依赖的所有库和支持构造都已经正确地安装好了。软件包还可以在安装期间的各个不同时刻运行脚本，因此它们能做的事情不仅限于提供新文件。（软件包卸载以后系统不能恢复原状的大部分情况都可能归结为这个特性。）

软件包还是一种发布用户自己本地化配置的很好途径。用户很容易就能创建一个这样的软件包：在安装它的时候，它会读取机器的本地化信息（或者从中央数据库获得这样的信息），用得到的信息生成本地的配置文件。您还可以把本地应用软件连同依赖关系一同打包，做成软件包，或者把没有按软件包格式正常发布的第三方应用做成软件包。您可以给自己的软件包设定版本，在安装自己新版的本地化软件包的时候，利用依赖关系机制自动升级系统。

您还能利用依赖关系机制创建软件包组。例如，可以创建一个软件包，它自己什么也不安装，但却依赖许多别的补丁。打开依赖关系，安装这个软件包，就能一步把所有补丁都安装好。

11.3.1 软件包管理系统

常用的软件包格式有两种。Red Hat、Fedora、SUSE 和其他几种发行版本使用 RPM，即 Red Hat Package Manager（Red Hat 软件包管理器）。Debian 和 Ubuntu 使用一种不同的**.deb** 格式。这两种格式的功能类似。

利用从 kitenet.net/programs/alien 获得的 **alien** 这样的工具，很容易就能在两种软件包格式之间进行转换。**alien** 对软件包内的软件一无所知，如果软件包的内容和发行版本不兼容的话，那么 **alien** 毫无用处。一般来说，最好坚持使用发行版本自己的软件包机制。

[3] 但要注意，**/dev** 可能已经在内核所在的文件系统里了。

www.TopSage.com

RPM 和.deb 这两种打包系统现在都是两层的全能配置管理工具。下面一层是安装、卸载和查询软件包的工具：RPM 的工具叫 **rpm**，**.deb** 的工具叫 **dpkg**。

在这些命令之上的是知道如何在 Internet 上找软件包、分析软件包间依赖关系、以及升级系统上所有软件包的系统。在这个层次上主要的竞争软件有：配合 RPM 系统运行的 **yum**，Red Hat Linux 专门的、也使用 RPM 的 Red Hat Network，还有 Debian 的 APT（Advanced Package Tool，高级软件包工具）。APT 源自于**.deb** 的世界，但是现在能很好地平等处理**.deb** 和 RPM 软件包。

下面我们会看看底层的 **rpm** 和 **dpkg** 命令。我们将在 11.4 节讨论在这些底层工具上建立的全面的更新系统（例如 APT 和 **yum**）。

11.3.2 rpm：管理 RPM 软件包

rpm 命令安装软件包、核实和查询软件包的状态。它以前也可以构造软件包，但是这项功能现在已经分拆出来交由 **rpmbuild** 命令负责。不过，**rpm** 的选项仍然有复杂的相互关系，它们只能按某些组合一起使用。最好把 **rpm** 想成是碰巧有相同名字的几个不同命令。

用户告诉 **rpm** 进入的模式（比如**--install** 或者**--query**）指出了用户希望访问到 **rpm** 多种特性中哪一个。命令 **rpm --help** 按模式分别列出了所有选项的清单，但是如果要频繁和 RPM 软件包打交道，那么值得花点时间稍微细读一下 **rpm** 的手册页。

rpm 的基本选项有：**--install**、**--upgrade**、**--erase** 和**--query**。**--query** 选项稍微有点复杂，因为它只用于启用其他选项；必须再给一个命令行标志来提出特定的问题。例如，命令 **rpm --query --all** 会显示出系统已经安装的软件包的完整清单。

让我们看一个例子。假定您因为最近发布了有关 OpenSSH 的一个新的安全补丁，需要安装它的新版本。一旦您已经把软件包下载到了本地的计算机上，那么只要一条命令 **rpm --upgrade** 就用新版本替换了老版本：

```
# rpm --upgrade openssh-2.9p2-12.i386.rpm
error: failed dependencies:
openssh = 2.9p2-7 is needed by openssh-askpass-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-askpass-gnome-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-clients-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-server-2.9p2-7
```

或许还没这么简单。我们在这儿看到，有若干其他软件包需要当前安装的 OpenSSH 版本，即 2.9p2-7。**rpm** 命令不让我们升级 OpenSSH，因为改动可能会影响到别的软件包。始终都会发生这样的冲突，这也就是开发类似 ATP 和 **yum** 这样的系统的主要动机。在实际中，我们不必手工搞清依赖关系，而是继续只用 **rpm** 达到本例的目的。

我们可以用**--force** 选项强行升级，但那往往是个糟糕的想法。这不是 Windows，有了依赖关系的信息并不会挡道，而是会节省时间和精力。没有比因为升级使 SSH 连不上远程系统这样的事情，更能毁掉系统管理员一个上午的时间。

相反，我们也要找到被依赖的软件包的更新版本。如果我们聪明的话，在尝试升级之前就已经能判断出会有其他软件包依赖 OpenSSH：

```
# rpm --query --whatrequires openssh
openssh-askpass-2.9p2-7
openssh-askpass-gnome-2.9p2-7
openssh-clients-2.9p2-7
openssh-server-2.9p2-7
```

现在假定我们已经得到了所有软件包更新后的副本。我们可以一次一个地来安装它们，但是 **rpm** 足够聪明，我们能一次把它们全装上。如果在命令行列出多个 RPM，**rpm** 会在安装它们之前根据依赖

www.TopSage.com

大家网 TopSage.com

关系对它们进行排序。

```
# rpm --upgrade openssh-*
```

酷！看上去它成功了，确认一下：

```
# rpm --query openssh
openssh-2.9p2-12
```

注意，即使我们没有给出软件包完整的名字或者版本，**rpm** 也能明白我们所说的是哪个软件包。

11.3.3 dpkg：管理 Debian 的软件包

就像 RPM 软件包有个全能的 **rpm** 命令一样，Debian 软件包也有 **dpkg** 命令。**dpkg** 比较有用的选项有：**--install**、**--remove**、以及**-l**，它会列出您的系统上已经安装的软件包。注意，**dpkg --install** 安装软件包之前会把系统上已经有的老版本删除掉。

假定 Debian 的安全小组最近发布了一个针对 **nvi** 的修正，给某个潜在的安全问题打上了补丁。在得到这个补丁之后，我们运行 **dpkg** 安装它。由此可见，它显示的信息要比 **rpm** 更多，准确地告诉了我们它所做的事情。

```
# dpkg --install ./nvi_1.79-16a.1_i386.deb
(Reading database ... 24368 files and directories currently installed.)
Preparing to replace nvi 1.79-14 (using ./nvi_1.79-16a.1_i386.deb) ...
Unpacking replacement nvi ...
Setting up nvi (1.79-16a.1) ...
Checking available versions of ex, updating links in /etc/alternatives ...
(You may modify the symlinks there yourself if desired - see 'man ln'.)
Leaving ex (/usr/bin/ex) pointing to /usr/bin/nex.
Leaving ex.1.gz (/usr/share/man/man1/ex.1.gz) pointing to
    /usr/share/man/man1/nex.1.gz.
...
```

我们现在可以用 **dpkg -l** 来看看安装是否起作用了。**-l** 标志接受一个可选的搜索模式，所以我们可以只搜索 **nvi**：

```
$ dpkg -l nvi
Desired=Unknown/Install/Remove/Purge
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version          Description
+++-==============-================-================================
ii  nvi            1.79-16a.1       4.4BSD re-implementation of vi.
```

我们的安装操作似乎已经顺利完成了。

11.4 高级软件包管理系统

APT、yum 和 Red Hat Network 这样的基础软件包管理系统目标都相同：

- 简化定位和下载软件包的过程；
- 自动进行系统更新或者升级；
- 方便管理软件包间的依赖关系。

www.TopSage.com

显然，这几种系统所面对的不仅仅是客户端的命令。它们都要求发行版本的维护人员以一种统一的方式组织好提供的软件，这样客户机就能访问和推断软件。

既然没有一家提供商能包含整个“Linux 的软件世界”，所以这几种系统都允许有多个软件库。软件库可以就在本地网络，这些系统为营造用户自己的内部发布系统提供了一个良好的基础。



Red Hat Network 同 Red Hat Enterprise Linux 紧密地结合在了一起。它是一种要付费的商业服务，而且在吸引人的 GUI 和自动化的能力方面，提供了比 APT 和 **yum** 更多的功能。遗憾的是，在这些表面现象之下掩盖着相当神秘的黑箱。它的客户机能够访问 **yum** 和 APT 的软件库，这种能力使得 CentOS 这样的发行版本可以让客户端的 GUI 适用于 Red Hat Network 之外的其他软件库。

APT 比 Red Hat Network 有更好的文档说明，移植性更强，而且是免费的。在您可以用它来做什么这方面，它也更为灵活。APT 源自于 Debian 和 **dpkg**，但是它已经扩展到能够支持 RPM，在我们举例的发行版本上，都有合适的 APT 版本可用。对于软件发布来说，APT 是目前我们可以有的最接近通用标准的工具。

yum 的功能类似 APT，但它专门用于 RPM。尽管如果把它指到格式正确的软件库上，它能在任何基于 RPM 的系统上运行，但它默认只是 Fedora 的软件包管理工具。

在做比较的时候，人们一般把 **yum** 当做是 APT 用于 RPM 的版本，但这么认为并没有明确有说服力的技术原因。**yum** 在 RPM 世界的根基更深，这让它成为领头羊也在情理之中。在 2005 年的早些时候，APT-RPM 的最初开发者 Gustavo Niemeyer 放弃了它，转而去开发一个更为复杂的系统（labix.org 的 Smart Package Manager，这个软件包管理工具还没有成为主流，但人们普遍期望它是下一个重大软件），这给 APT-RPM 的未来蒙上了一层阴影。最后 Panu Matilainen 接手继续开发 APT-RPM，这个软件项目的开发工作目前再度活跃起来。

我们喜爱 APT，如果您想要建立自己的软件包自动发布网络的话，不管您的站点当前在用的 Linux 发行版本是怎样的，APT 都是一个供您考虑的可靠选择。



SUSE 在软件包管理领域做得并不好。它虽然使用 RPM 包，但是以前的发行版本只支持通过 SUSE 自己的 YaST Online Update 工具做系统升级。经过一番兼容并包之后，SUSE 最近已经在某种程度上增加了对 **yum**、APT-RPM 和 Novell 自己的 ZENworks Linux Management 代理的支持，而 ZENworks 成为 SUSE 首要的软件更新管理工具（了解命令行界面可参见 **rug**）。

ZENworks 属于更大的一个产品线，这个产品线体现出 Novell 要在跨平台配置管理领域里占据主导地位的努力。它是您最好的选择吗？如果您采用了 Novell 的产品，而且对付费支持感兴趣的话，或许能算吧。那些想要保持软件包的管理能够免费而且没有倾向性的站点，可能会研究前面提到的 Smart Package Manager。

11.4.1 软件包的库

Linux 的各个发行版本都维护着自己的软件库，这些软件库和发行版本选择的软件包管理系统携手工作。软件包管理系统的默认配置通常指向一个或者几个由发行商控制的知名 Web 或 FTP 服务器。

不过，这样的软件库应该包含什么内容却并不那么显而易见。它们应该只包括正式的、主要发布的软件包吗？还是正式发布加上当前的安全更新？还是正式发布中所有软件包的最新版本？没有得到发行方的官方支持，但却有用的第三方软件该有吗？该有源代码吗？该有针对多种硬件体系结构的二进制软件吗？当运行 **apt-get upgrade** 或者 **yum upgrade** 更新系统的时候，到底应该做些什么？

www.TopSage.com

一般来说，软件包管理系统必须解决上述所有问题，而且必须让站点很容易选择出自己规定的专门交叉集合，以此作为自己的“软件圈子”。下面的概念有助于理清这个过程的脉络。

- “发布（release）”是对软件包的世界所做的一个前后一致的快照。在 Internet 时代之前，有名有姓的操作系统都多少不怎么容易变化，某个时期就是一个固定的版本，安全补丁则单独提供。现如今，发布是一个更加模糊的概念。发布随着软件包的更新而不断演变。有些发布，比如 Red Hat Enterprise Linux，特意做成发展变化很缓慢。在默认情况下，这类发布只融入安全更新。而另外一些发布，比如 beta 版发布，变化频繁而且幅度大。但是在所有情况下，发布都是基线，是目标，是“我想更新我的系统，让它看上去像的那个东西”。
- “组件（component）”是一次发布里的软件子集。各个发行版本对自身的划分不尽相同，但是它们却都会明确区分核心软件和额外软件，核心软件由发行商做保障，而额外软件则由更为广泛的社群来提供。在 Linux 界还有一种区分，这就是在发布里，有开放源代码的部分和受某种限制性许可证协议影响的部分。

从系统管理的角度来看，特别要注意的是仅仅包含安全补丁的那些最稳定的软件组件。有些发布可以把一个安全组件同一个稳定不变的基线组件组合起来，创建出一个相对稳定的发行版本。

- “体系结构（architecture）”代表一种特定类型的硬件。希望属于一种体系结构的机器都很相似，它们都能执行相同的二进制代码。发布针对不同的体系结构有专门的版本，例如，“Fedora Core 5 for the i386 architecture（针对 i386 体系结构的 Fedora Core 5）”。因为发布可以细分为若干组件，所以每个组件也对应各种体系结构有专门的版本。
- “单个软件包”是组成组件的基本元素，因此它们也间接成为组成发布的基本元素。软件包通常特定于某种体系结构，其版本与主发布无关，也和其他软件包没有关系。软件包和发布间的对应关系由网络上软件库的设置方式暗中决定。

有些软件组件并不是由发行版本的提供方来维护的（例如，Debian 的“contrib”或者 Fedora 的“extras”），这就带来了问题，这些软件组件同核心 OS 发布的关系如何确定。它们真的能够叫做特定发布的“一个组件”吗？从管理软件包的角度来看，答案很清楚：extras 确实算组件。它们和一个特定的发布有关系，而且它们随发布一起变化。从系统管理的立场来说，对这类软件包的控制独立于发布要引起注意，但是它不会影响到软件包的发行体系。

11.4.2 RHN：Red Hat 网络

随着 Red Hat 远离消费类的 Linux 业务，Red Hat Network（RHN）已经成为 Red Hat Enterprise Linux 的系统管理平台。您要通过订购方式购买对 RHN 的访问权。在最简单的层面上，您只要把 Red Hat Network 当作一个色彩斑斓的 Web 门户网站和邮递列表来用就行了。按照这种方式来用它的话，它同各个 UNIX 厂商多年来所支持的补丁通知邮递列表没有太大的区别。但是如果您愿意花钱的话，会享受到更多的功能。

Red Hat Network 提供了一种非常漂亮的 GUI 界面，供下载新软件包使用，它还提供了一种命令行的备选方式。它甚至可以在不需要人工干预的情况下，下载和安装新软件包。一旦您取得了注册，您的机器就会得到它们所需要的全部补丁和 bug 修正，而您不必放下手中的 Quake 游戏。自动注册的不利一面是要由 Red Hat 来决定您需要什么样的补丁。您可能要考虑您到底对 Red Hat（以及被他们打包的软件产品的维护方）有多大的信任程度，相信他们不会把事情搞糟。鉴于过去在碰到比如应该提供哪种编译器这样的小事情时，Red Hat 所做的一些有趣的选择，有些人可能会持怀疑态度。

一种合理的折衷方法可能是，让单位内的一台机器注册自动更新功能。您可以定期从那台机器得到快照，按内部发布的可能候选来进行测试。不过，在着手这么做之前，要先研究一下 Red Hat 的许可证协议（从 www.redhat.com/licenses 可以获得）。您可能会对下面的内容感到吃惊：Red Hat 称其对

www.TopSage.com

于通过 RHN 发布的开放源代码软件享有自己的权益，但却没有提您已经同意让 Red Hat 随意审查您的系统。

11.4.3 APT：高级软件包工具

APT（Advanced Package Tool，高级软件包工具）是最成熟的软件包管理系统。用一条 **apt-get** 命令就可以更新整个一个系统的所有软件，甚至（就和使用 RHN 一样）还可以不需要人工干预，保持您的机器不断得到更新。

因为 APT 源自于 Debian 的世界，所以它原本只支持**.deb** 格式的软件包。不过，现在 APT 已经被移植到了 RPM 软件包的机制上。这个版本叫做 APT-RPM，可以从 apt-rpm.org 得到它。本章后面将会介绍 **yum** 系统，它提供了类似的功能，但却基于 RPM，Fedora 发布直接支持 **yum** 系统。在 **yum** 和 APT-RPM 之间进行选择的时候，您的站点和发行版本内部侧重的不同，要比系统间技术上的差异重要得多。要采用得到很好支持和简便易行的方式。

在 Debian 系统上使用 **apt-get**（以及实际上 Debian 软件包的所有管理工作）的第一条原则就是忽略 **dselect** 的存在，它充当了 Debian 软件包系统的前端。虽然它并不是个坏想法，但是用户界面做得太差了。Debian 的文档将会试着指导您使用 **dselect**，但您自己要挺住。

如果您采用 **apt-get** 来管理从标准镜像站点做的 Debian 或者 Ubuntu 安装，要看看能得到什么软件包，最简单的办法就是访问 packages.debian.org 或者 packages.ubuntu.com 这两个站点。两种发行版本都提供了一个很有用的搜索界面。如果建立了自己的 **apt-get** 服务器，那么您当然会知道能得到什么软件包，也能以您愿意的任何方式把它们给列出来。

各种发行版本一般都包含几个空的软件包，之所以存在这样的软件包，只是为了把其他软件包当作前提条件列出。因为 **apt-get** 会根据需要，自动下载和升级作为前提的软件包，所以这样的软件包就能很容易地把几个软件包当作一个整体来安装或者升级。例如，安装 **gnome-desktop-environment** 这个软件包就会保证您获得安装和运行 GNOME 用户界面所需的全部软件包。

一旦您已经建好了自己的 **sources.list** 文件，而且知道您想要的软件包的名字，剩下来的唯一任务就是运行 **apt-get update** 来刷新 **apt-get** 缓存的软件包信息。在那之后，只需运行 **apt-get install** *package-name* 来安装软件包就行了。同样的命令也会更新已经安装过的软件包。

假定我们想要安装 **sudo** 软件的一个新版本来修正一个安全 bug。首先，执行一次 **apt-get update** 总是比较明智的：

```
$ sudo apt-get update
Get:1 http://http.us.debian.org stable/main Packages [824kB]
Get:2 http://non-us.debian.org stable/non-US/main Release [102B]
...
```

现在我们实际上可以获取软件包了。注意，我们在我们获取新的 **sudo** 软件包的同时也正在使用 **sudo**——**apt-get** 甚至能在软件仍在使用的时候升级它们！

```
$ sudo apt-get install sudo
Reading Package Lists... Done
Building Dependency Tree... Done
1 packages upgraded, 0 newly installed, 0 to remove and 191 not upgraded.
Need to get 0B/122kB of archives. After unpacking 131kB will be used.
(Reading database ... 24359 files and directories currently installed.)
Preparing to replace sudo 1.6.1-1 (using .../sudo_1.6.2p2-2_i386.deb) ...
Unpacking replacement sudo ...
Setting up sudo (1.6.2p2-2) ...
Installing new version of config file /etc/pam.d/sudo ...
```

www.TopSage.com

11.4.4 配置 apt-get

配置 **apt-get** 相当直观易懂，需要您了解的每样东西都可以在 *APT HOWTO* 里找到：http://www.debian.org/doc/manuals/apt-howto。

apt-get 最重要的配置文件是/etc/apt/sources.list，它告诉 **apt-get** 到哪儿去找它的软件包。这个文件里每行规定的内容如下：

- 软件包的类型，目前 Debian 类型的软件包用 deb 或者 deb-src，RPM 用 rpm 或者 rpm-src；
- 指向一个文件、CD-ROM、HTTP 服务器或者 FTP 服务器的 URL，从那里可以能够取得软件包；
- “发布（实际上是一个发布的名字）”，能让您提供软件包的多个版本。发行版本提供方使用这个字段来给自己的发行版本指定版本，但是您也可以将它用于您的内部发行系统；
- 组件可能的清单——其实是发行版本内的软件包分类。

除非想要建立自己的 API 库或者缓存，否则默认配置一般就够用了。如果您有一条比较好的网络连接，那么应该注释掉用来指定 Debian 安装光盘的那些行。如果您想要下载源代码，就不要注释掉指定 `deb-src` 或者 `rpm-src` 的那些行。

只要您正好在编辑这个文件，就应该把镜像的标识改为离您最近的一个，在下面的地址可以找到完整的 Debian 镜像列表：http://www.debian.org/misc/README.mirrors。



Ubuntu 在 wiki.ubuntu.com/Archive 维护了一个类似的列表。

要让事情更简单，Debian 提供了一个叫做 **netselect-apt** 的工具，它自动会替您生成一个 **sources.list** 文件，它根据 ping 的时间，选择它能找到的最近的映像站点。**netselect-apt** 是 **netselect** 软件包的一部分，您可以从离您最近的 Debian 镜像站点得到它（现成的 **netselect-apt** 或多或少和 Debian 的镜像系统有联系，但是大多数软件包也能在 Ubuntu 上用，没有问题）。

您也应该确保把 security.debian.org 或者 security.ubuntu.com 作为一个来源列出，以便您能访问到最新的安全补丁。

11.4.5 /etc/apt/sources.list 文件的例子

下面的例子不但将 http.us.debian.org 用于稳定的存档文件，而且加入 non-us.debian.org 作为美国之外使用的软件包（不能从美国出口的加密软件，以前很重要，现在已经没那么重要了）的来源。我们已经加上了 security.debian.org，它是所有安全补丁的来源，还加上了我们本地的 APT 服务器 local-debian-server。最后，我们打开了下载源代码的功能。

```
# General format: type uri distribution [components]
deb http://ftp.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
deb http://local-debian-server/mypackages/ ./
deb-src http://http.us.debian.org/debian stable main contrib non-free
deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src http://security.debian.org stable/updates main contrib non-free
```

其中 *distribution* 和 *compoments* 字段帮助 **apt-get** 遍历 Debian 软件库中文件系统的层次结构，这个结构的布局有标准可循。根发行版本既可以是最近的主流发行版本（`stable`），也可以是当前正在开发的发行版本（`unstable` 或者 `testing`），或者是一个特定发布的名字（例如 `etch`）。能够访问

www.TopSage.com

到的组件一般有 main、contrib 和 non-free。

这个例子使用的是稳定的软件库。在实际中间，稳定的发行版本没有别的发行版本那么频繁地变化。所有最新的软件包都包含在 Debian 最新但尚不完善的不稳定（unstable）发行版本中。“不稳定”未必意味着软件包自身不稳定，而是整个发行版本的组成不稳定。每周更新的软件量一般超过100MB。

sources.list 文件里的配置行按顺序逐行分析，所以从理论上说，您可以把 unstable 和 testing 行放在文件末尾，让 stable（稳定）版本优先。这个方法的问题是，因为 APT 的依赖性延伸，一个不稳定的软件包会把它所依赖的所有软件包的不稳定更新版本都带进来。这些软件包接下来又可能会把自己所关联软件的不稳定版本再带进来，以此类推。一个老鼠坏了一锅汤，不要在您的生产系统上安装不稳定的软件包。

如果您必须把一个从 unstable（不稳定）发行版本来的软件包加到您的生产环境中，正确的做法是使用一个“向后移植（backport）”的软件，即在稳定发布上重新编译它，让它同稳定的库链接。为了找到这些 backport 和其他细节，可检查位于 www.apt-get.org 的 APT 搜索引擎。在 Norbert Tretkowski 的站点 www.backports.org 上，能够找到许多向后移植的软件包（不只是给出链接）。在这个软件库中的向后移植软件包质量都很高，对外部的依赖性也最低。

11.4.6 使用代理扩展 apt-get

如果您打算在大量主机上使用 **apt-get**，或者想要把软件包缓存在本地——为每台机器都下载每个软件包的一个副本对带宽的使用来说并不划算。您可能也需要指示 **apt-get** 通过代理下载软件包，如果防火墙要求有这层保护作用的话。

既然 **apt-get** 用的就是普通 HTTP 和 FTP 协议，所以您可以使用任何碰巧已经装好的 Web 代理。**apt-get** 认可环境变量 http_proxy，同样可以在文件**/etc/apt/apt.conf** 里显式地设置代理：

```
Acquire::http::Proxy "http://proxyserver:8080/
```

替代普通 Web 代理的一种方法是采用一个叫做 **apt-proxy** 的小程序。虽然它的名字叫代理，但是却不是一个真正的代理，而只是一个程序，从真正的 APT 服务器那里用 **rsync** 得到软件包，建立它们的缓存。**apt-proxy** 可以从下面的地址得到：http://sourceforge.net/projects/apt-proxy。

11.4.7 设置内部 APT 服务器

除了使用代理，也可以建立您自己管理的 APT 服务器，并且把内部的客户机都指向它。这种模式能让您调整向客户机提供的软件包、轻而易举地以推的方式进行升级（只在服务器上安装新版本）、把自己的应用作为软件包来发布，而且最重要的是，提供自己的发行版本。

既然 **apt-get** 使用标准的协议（HTTP 和 FTP）下载它的软件包，建立一台 APT 服务器所要做的全部工作就是建立一个 Web 或者 FTP 服务器，提供适当的内容[4]。既然与 HTTP 相关的服务器和工具得到了广泛的使用，所以 HTTP 可能是配合 APT 使用的最方便的协议。21.2 节里给出了建立 Apache Web 服务器的指导。

服务器上的软件包可以都放在一个目录里，或者它们也可以像 Debian 和 Ubuntu 的镜像站点那样分布在一个层次型目录结构里。

除了提供软件包的文件之外，还必须生成两个软件包的汇总文件：**Packages.gz** 和 **Contents.gz**。

[4] 另外还有别的做法。例如，您可以设置一台 NFS 服务器，上面有更新文件，或者刻录一张 DVD 传给每个系统使用。

www.TopSage.com

前者是服务器上软件包及其依赖关系的清单，并且用 **gzip** 压缩过了。**apt-get update** 使用这个清单来确定能够获得哪些补充的软件包。后者建立原始文件到包含它们的软件包的映射关系，**apt-get** 本身实际上并不会用到这个文件。**apt-utils** 这个软件包中包含的 **apt-ftparchive** 命令会自动替您生成这两个汇总文件。

一旦已经创建好了汇总文件，剩下来的工作就很容易了。在客户机上的**/etc/apt/sources.list** 文件中像下面这样的一行会把 **apt-get** 连到您的本地服务器上。

```
deb http://local-server/mypackages/ ./
```

在每台客户机上运行 **apt-get update**，然后照常使用 **apt-get**。

如果既要发布源代码，也要发布二进制软件包，只要把源代码软件包放到服务器上即可。Debian 发布的源代码有 3 个部分：普通的**.tar.gz** 文件、一个可选的**.diff.gz** 文件（供软件包维护程序去显示它们相对于代码原有版本所发生的变化）、一个**.dsc** 文件（其中包含软件包的说明），这是和 RPM 不一样的，RPM 有一个对等的 SRPM，作为源代码软件包。与 **Packages.gz** 等价的源代码是 **Sources.gz**，它也是由 **apt-ftparchive** 生成的。

前面 **sources.list** 文件的例子里没有指定“发布（distribution）”参数。如果您想要用自己的发行版本的名字，作为内部制订版本的形式，就把每个版本放入一个子目录，把 **sources.list** 文件中的./改为版本名或者版本号。

类比 Debian 自己的“stable”和“testing”发布来创建通称的发行版本，比如“test”和“production”，这种做法往往用处不小。您可以在服务器上用符号链接，把这些名字指向特定的发布，以后只要改变链接的目标目录，就能重新规定发布的内容。例如，当您对一个 test（测试）发布感到满意，觉得它能部署它了，就可以把“production（生产）”的符号链接指到同一个目录。客户机把自己与 production（生产）发布进行同步，自动获得变更数据。

11.4.8 自动执行 apt-get

您可以从 **cron** 定期执行 **apt-get**。即使您并不打算自动安装软件包，也可能想要定期执行 **apt-get update** 来保持包汇总信息得到更新。

apt-get dist-upgrade 会下载并安装已经在本地机器上安装过的任何软件包的新版本。**dist-upgrade** 类似于 **upgrade**，但是在处理依赖关系上稍微聪明一点儿。**dist-upgrade** 可能想要删除它认为和更新过的系统不兼容且解决不了的一些软件包，所以要对可能出现意想不到的结果有所准备。

如果您真的想要冒险，那么就让机器自动执行这项升级——使用**-yes** 选项，这样就无需人来干预操作，它将对 **apt-get** 可能回答的任何确认性问题都热诚地回答说“Yes（是）!”。

直接从发行版本的镜像站点执行自动升级操作或许并不是个好主意。不过，如果您有了自己的 APT 服务器、软件包和发布控制系统，那么这就是保持客户机同步的妥善方式。像下面这样的一个简短的 shell 脚本能够让一台机器保持和它的 APT 服务器同步更新。

```
#!/bin/sh
apt-get update
apt-get -yes dist-upgrade
```

如果想要在夜间运行这个脚本，就从 **cron** 来调用它。您还可以在一个系统启动脚本中调用它，让机器在引导时更新。参考第 8 章了解有关 **cron** 的更多知识，参考第 2 章了解有关启动脚本的更多信息。

如果您在许多机器上从 **cron** 里执行更新，那么应该让时间岔开，以确保每次不会让每台机器都更新。17.2.3 节末尾给出的简短的 Perl 脚本就能帮助完成这项任务。

www.TopSage.com

如果不能完全信任软件包的来源，那么就考虑采用自动下载所有改动过的软件包，但并不安装它们这种方式。用 **apt-get** 的**--download-only** 选项来达到这种效果，然后手工复查软件包，并且安装您想要更新的软件包。下载的软件包都保存在**/var/cache/apt** 里，随着时间的推移，这个目录会变得相当大。用 **apt-get autoclean** 命令可以从这个目录中清理掉没有用的文件。

如果您使用了一个正规设计的稳定的 Debian 发行版本，那么我们将毫无保留地建议您采用自动更新。稳定发行版本中的变化一般限制在安全更新上，而且完整性测试做得很好。唯一可能发生的问题是您或许在一个新的主发布出现的时候不想自动升级。为了避免发生这样的问题，可以在 **sources.list** 文件里明确指定发行版本的名字而不是 stable 这个关键字。

11.4.9 yum：管理 RPM 的发布

yum（Yellowdog Updater，Modified）是基于 RPM 的元软件包管理器[5]。把 **yum** 叫做 **apt-get** 的克隆可能有一点儿不公平，但是它从提法上和实现上都类似于 **apt-get**，只是实际上用起来更简洁和速度更慢。**yum** 是 Fedora 的正式软件包管理系统，在许多别的发行版本上也预装了它。如果有必要，您可以从 linux.duke.edu/yum 获得它的最新版本。

和 **apt-get** 一样，服务器端的命令（**yum-arch**）汇编了一个数据库，其中是大量软件包（往往是一个发布中的软件包）的头信息。然后通过 HTTP 或者 FTP 把头信息数据库随软件包一起对外共享。客户机使用 **yum** 命令推断出依赖性限制，然后完成安装所需软件包要求的其他任何工作。如果要安装的软件包依赖别的软件包，那么 **yum** 也会下载并安装那些软件包。

apt-get 和 **yum** 之间的相似性也扩大到了它们所支持的命令行参数上。例如，**yum install foo** 会下载和安装软件包 **foo** 的最新版本（如果必要的话，还有其依赖软件包的最新版本）。但两者还是至少有一点截然不同之处：**apt-get update** 会刷新 **apt-get** 的软件包信息缓存，但是 **yum update** 会更新系统上的每个软件包（它类似于 **apt-get upgrade**）。让情况更混乱的是，还有 **yum upgrade** 命令，它的作用和 **yum update** 命令相同，区别在于 **yum upgrade** 相当于一次大规模的升级，连带了旧的要淘汰的包也升级（译者注：**yum upgrade** 相当于 **yum update --obsoletes**）。

yum 不匹配不完整的软件包名，除非您在调用 **yum** 时使用了 shell 的通配符（比如*和?）明确要求这样做。例如，**yum update 'perl*'**会把所有名字以“perl”开头的软件包都更新。要记住通配符要用引号引起来，让 shell 不会去妨碍它们。

和 **apt-get** 不一样，您每次运行 **yum** 的时候，它默认要把自己的软件包信息缓存和网络软件库的内容对照进行验证。使用**-C** 参数可以不做验证，**yum** 让缓存更新本地缓存（运行要花一些时间）。遗憾的是，**-C** 参数并不能让 **yum** 的迟缓性能有多大改善。

yum 的配置文件是**/etc/yum.conf**。它包含通用选项和指向软件包库的指示。一次能够有多个活动的软件库，每个软件库都可以关联多个 URL。

Fedoar 在它的标准发行版本树中包含 **yum** 格式的元数据，所以 Fedora 的用户可以随意把他们的 **yum.conf** 文件指向最近的镜像进行更新。RHEL 在这方面没有这么做，或许为了促进对 RHN（Red Hat Network）的订购。

11.5 版本控制

生活中总会犯错。因此，跟踪您所做的改动就变得很重要，这样一来，当改动造成问题的时候，

[5] 不要同 yum3x.com 的 Yum Fish Bait with Live Prey Technology (LPT)混淆了（译者注：这是一家提供鱼饵的公司）。

www.TopSage.com

您就可以轻而易举地恢复到一个已知是好的配置上。如果您的站点使用了一种正式的版本化了的发布系统，那么回滚（rollback）工作就会变得很容易，但这是一种重量级的解决方案，只能粗略地进行控制。如果您只需要改一两个文件该怎么办呢？

在这一节里，我们讨论一些以单个文件为单位管理修改的常用方法。这些方法是对内部发布控制等其他较大改动的有力补充——选择同您的本地需要和您站点复杂程度相符合的一类工具。

11.5.1 创建备份文件

给您修改的文件留备份是一项历史悠久的管理传统，要依靠本地化脚本以及系统管理员个人来遵守这个传统。备份文件能让您退回到配置的一个早先状态，但或许更为重要的是，它们还能让您把一个文件当前和过去的版本做差异比较，得知做了哪些改动。

您最好通过把原来的文件（用 **mv** 命令）改为一个新名字（比如 filename.old 或者 filename.bak），然后再把它拷回一个用原名的文件，创建备份文件。使用 **cp** 的**-p** 选项保留文件的属性设定。一旦您更新为这个文件的活动版本，它的修改时间就反映出了最近的修改，而备份文件的修改时间则反映了上一次修改的时间（如果您只用 cp 拷贝，没有带-p 选项，那么两个文件的修改时间则接近）。把原来的文件移动到一边的做法也能处理好这样的情形，即一个活动进程打开对该文件的访问：您对文件的当前活动拷贝所做的修改，要等到该文件被关闭再打开之后才看得到。

系统定期被备份到磁带上的系统仍然能够从手工创建的备份文件上获益。从一个备份文件恢复要比从一卷磁带恢复快得多，也容易得多，手工备份多保留了一层历史信息。

11.5.2 正式的版本控制系统

备份文件非常有用，它们在小站点上实际效果最好。复杂性和健壮性再高一个层次的做法是采用正式的版本控制系统（revision control system），这种系统是一种软件包，能够跟踪、保存文件的多个版本，并且可以提供访问这些不同版本的手段。这类版本控制软件包源自于软件开发领域，但是它们也对系统管理员非常有用。

版本控制系统解决了几个问题。首先，它们提供了一种有组织的方式来跟踪对一个文件的修改历史，从而让人理解改动文件的来龙去脉，并且能够恢复早先的版本。其次，它们把版本的概念扩展到了单个文件的层次之外。有关系的一组文件可以按其相互间的依赖性一起指定版本。最后，版本控制系统能够协调多个编辑人员之间的活动，使得竞争条件（race condition）不会让任何人的改动永久丢失[6]，这样一来，多个编辑人员做出的不兼容的改动不会同时起作用。

常用的最简单的版本控制系统叫做 RCS，名字正好就是版本控制系统（Revision Control System）。RCS 已经存在了数十年，许多系统上都预装了它。

另一种选择是一种称为 CVS（Concurrent Versions System，并发版本系统）的开放源代码系统，它在 RCS 之上增加了一些功能。它支持一种分布式的模型（配合远程服务器来使用），而且更好地支持了多位开发人员的协同工作。有大量的站点已经将 CVS 用于系统管理的任务，它的客户机/服务器的功能特别有用。遗憾的是，CVS 有一些概念上的缺陷，使得它毁誉参半。

在开放源代码领域新近出现（但仍然需要时间考验）的一种版本控制系统是 Subversion，它提供了 CVS 所有的优点，但它似乎有更合理的默认举动。从系统管理的角度来看，它的主要缺点在于项目

[6] 例如，假定系统管理员 Alice 和 Bob 都要编辑同一个文件，而且假定他们两个人都做了一些修改。Alice 先保存这个文件。当 Bob 再保存他的文件副本的时候，就覆盖了 Alice 的版本。如果 Alice 已经退出了她的编辑器，那么她所做的改动就彻底丢失了，再也恢复不了了。

www.TopSage.com

的模型过于以目录为中心。不过，它仍然不失为一种非常好的系统，是可供系统管理使用的一种合理选择。

最近几年，我们目睹了开放源代码版本控制系统的繁荣发展，可供选择的系统扩大了几乎一个数量级。在比较新的系统中，主要的竞争者有 Monotone、Darcs、Arch 和 BazaarNG。这些系统都很有意思，但是它们看上去都侧重于非集中式的工作：多个分支、多个库、众多的并行开发。根据我们的观点，传统的“集中库”模型更适合于系统管理方面的版本控制。

还有几种商业的版本控制系统。如果您在一家开发公司工作，那么您可能已经用过这几种系统中的一种，而且可能试着把它用于系统管理数据。不过，根据我们的经验，商业系统对于系统管理员使用来说，一般是大材小用了。

如果您是从头开始，我们建议一开始从 RCS 获得对版本控制的总体感觉。如果您想要建立一个集中式的系统管理信息库，那么应该用 Subversion。

11.5.3 RCS：版本控制系统

RCS（版本控制系统，Revision Control System）是现存的最古老的 UNIX 应用程序之一。它实际上是一种相当简单的系统。它在单个文件的层面上执行操作，并且把每个文件的版本信息保存在独立的隐蔽文件中。这个隐蔽文件的名字和原来的文件名一样，但是在原名后面加了.v 两个字符。例如，如果您把文件**/etc/syslog.conf** 置于 RCS 的控制之下，那么 RCS 会将它的各个版本保存在**/etc/syslog.conf.v** 里。

为了减少混乱，RCS 在和原来文件相同的目录下寻找一个叫做 **RCS** 的目录。如果有这个目录，RCS 就把.v 文件放到那个目录里面，而不是把它们留在原来文件的目录里。这样一来，列出的目录内容就会变得清楚多了，因为许多文件都可以共享 **RCS** 目录。这是一种很棒的功能，而且也是我们极力推荐的一种做法。

RCS 的基本概念就是，您“检出（check out）”您想要修改的文件，而“检入（check in）”您所做的改动，把它们保存起来。相应地，您确实需要知道的 RCS 命令只有几个：**ci** 用于检入、**co** 用于检出，而 **rcs** 执行各种维护工作。如果您使用的编辑器是 **emacs**，那么就要完全不必使用命令行工具了，因为 **emacs** 内置了对 RCS 的支持。

要让一个文件开始处于 RCS 的跟踪之下，可以像下面这样在这个文件上运行 **ci**：

```
# ci -u syslog.conf
RCS/syslog.conf,v <-- syslog.conf
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> This is the syslog configuration file.
```

-u 标志让 **ci** 立即以一种未上锁（不可编辑）的状态检出 **syslog.conf** 文件。如果您省略了这个标记，那么 **ci** 就会检入这个文件，然后删除原来的副本，这或许并不是您想要的做法。

您每次想要改变一个受 RCS 控制的文件时，都必须检出它，并用 **co -l** 命令对它上锁：

```
# co -l syslog.conf
RCS/syslog.conf,v --> syslog.conf revision 1.2 (locked)
done
```

这一操作告诉 RCS，您打算修改这个文件了，RCS 不会让其他任何人检出这个文件，直到您再把它检入为止。

RCS 在未上锁的文件上取消了写权限，提醒用户在正确检出它们之前，不要编辑它们。一种常见的错误是，把一个文件载入您的编辑器，进行修改，但直到编辑器拒绝保存所做的修改，才意识到需

www.TopSage.com

要把文件检出。要纠正这样的错误，只要暂停编辑器，或者启动另一个 shell 窗口，运行适当的 **co -l** 命令，然后再试着保存就可以了。您只要用 **chmod** 命令就能把文件改为可写，但这样做会破坏和弄乱 RCS。

从理论上说，RCS 上锁避免了两个不同的人同时修改文件的情况出现。在实际中，您必须是 root 才能修改系统文件，因此一旦某个系统文件以 root 权限被检出以后，有 **sudo** 权利的任何人都能修改它。不过，如果第二个系统管理员试图再用命令 **co -l** 的时候，RCS 会提示说已经有一个可写版本了。系统管理员应该形成习惯，在他们想要修改由 RCS 控制的文件时都必须尝试先检出它。文件可写意味着“住手！有别人已经检出这个文件了。”

您可能偶尔会发现有其他人已经改动了一个文件，并且让它留在了上锁状态，甚至更糟糕，这个人没管 RCS，没对文件上锁就做了修改。您可以用 **rcsdiff** 复查犯错误的人所做的修改，这条命令是一个能懂 RCS 的 **diff** 版本。例如：

```
# rcsdiff syslog.conf
===================================================================
RCS file: RCS/syslog.conf,v
retrieving revision 1.3
diff -r1.3 syslog.conf 4c4
< define(LOGHOST,moonbase)
---
> define(LOGHOST,spacelounge)
```

最后一招是您可以用命令 **rcs -u** *filename* 打破上锁。这条命令提示您输入对所采取的措施给予的解释，并且发邮件给前面对文件上锁的用户（遗憾的是，同时就是 root）。

一旦您对检出文件所做的修改感到满意以后，就可以用 **ci -u** 命令把它检入。RCS 会要求您提供一条注释，说明您刚才做的是什么。虽然人们倾向于跳过这步，或者写些诸如“做了一次修改”这样的无用信息，但是应该坚持加注释这个习惯。在两年以后，当您要弄清楚为什么您要做改动的原因时，有帮助的注释会救您一命的。

```
# ci -u syslog.conf
RCS/syslog.conf,v <-- syslog.conf
new revision: 1.3; previous revision: 1.2
enter log message, terminated with single '.' or end of file:
>> Started logging debug messages to track down SSH problem
>> .
done
```

您可以用 **rlog** 命令查看一个文件的版本历史信息：

```
# rlog syslog.conf
RCS file: RCS/syslog.conf,v
Working file: syslog.conf
head: 1.3
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv

total revisions: 3; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2002/01/10 00:44:58; author: adam; state: Exp; lines: +1 -0
Started logging debug messages to track down problem
```

www.TopSage.com

```
----------------------------
revision 1.2
date: 2000/07/19 08:23:10; author: evi; state: Exp; lines: +2 -0
Changed log destination to new logmaster
----------------------------
revision 1.1
date: 1998/03/14 11:13:00; author: matthew; state: Exp;
Initial revision
=============================================
```

如果您想要看看文件在换到新日志主机之前的样子，可以用 **co** 的**-r** 选项检出这个文件的 1.2 版：

```
# co -r1.2 syslog.conf
RCS/syslog.conf,v --> syslog.conf
revision 1.2
done
```

这条命令用老版本替换了当前的 **syslog.conf** 文件，所以要确保在您做完以后执行一次正常的 **co** 命令，否则您（和 **syslogd**）都可能会被搞糊涂！另一种做法是 **co -r1.2 -p syslog.conf**，它把所需版本的内容发送到标准输出。遗憾的是，没有办法把文件检出到一个不同的文件名里。决不要检出老版本的上锁副本（用 **co -l** 命令），因为这一操作会在版本树中创建分支。版本分支偶尔会在源代码上使用，但是对系统管理员的工作来说几乎没有用处，只需略过 RCS 文档中所有涉及它们的地方，让您的生活更简单些即可。

有关 RCS 的更多知识可以在下面的地址找到：http://www.cs.purdue.edu/homes/trinkle/RCS/。

11.5.4 CVS：并发版本系统

RCS 有一些显著的缺点。您在编辑文件的时候，必须小心不要和您的同事同时编辑文件。您必须遵守一组相当专门的操作步骤。您不能一次安全地修改几个文件，而又让系统不受没修改完的文件的影响。这些（以及其他一些）不足之处推动了设计下一代 CVS（并发版本系统，Concurrent Versions System）的工作，CVS 是目前在 UNIX 和 Linux 系统上使用最为广泛的版本控制系统。

CVS 背后的主要思想之一是项目文件（及其历史版本）都保存在一个中心位置。和在 RCS 系统中一样，您要检出要开始在上面工作的文件，然后做完修改之后把这些文件再次检入到代码库里。CVS 的优势在于，它没有“上锁”和“解锁”检出的概念，若干人可以同时检出并修改相同的文件（“并发”由此得名）。

既然 CVS 可以让多个用户修改同一个文件，它必须提供一种方法，在文件被再次检入代码库的时候能把多个用户的修改集成到一起。这种合并工作在正常情况下会自动完成，但只有文件是文本文件，而且各种不同的改动互相并不矛盾的情况下，才能顺利实现。如果不满足上述条件，那么 CVS 就要靠人工检入，手工解决冲突。如果对一个文本文件做的多个改动相互冲突，那么 CVS 就要插入帮助性的注释，反映出冲突出现的位置。

理论就说这么多。听上去挺酷的，实际也确实如此——即使合并工作也做得不错。但是 CVS 仍然存在一些问题。

- 它不支持“原子提交”。如果两个人都尝试检入影响到多个文件的一次重大修改，两个版本可能都以接受一半而告终，交给每个操作员解决的是随意选择的一些冲突。这可不对。
- 要改变一个文件的名字，必须先把它拷贝为新名字，然后删除原来的文件，因此失去了这个文件以前的所有历史信息。类似地，只有先拷贝目录，再删除原来版本的目录，才能改变目录的名字，除此之外别无他法。

www.TopSage.com

- 文件属性不受版本控制。文件第一次检入时的设定就一直保留下来不会变化。

尽管如此，许多开放源代码软件项目仍然使用 CVS。之所以如此，不是因为它无懈可击，而是因为以前缺乏好用的替代软件。然而，随着很多开发小组投入到开发 CVS 替代软件的项目上，这种情况已经发生了变化。现如今的选择多种多样，竞争的主旨可能落在了哪种版本控制系统应该超过 CVS 的问题上。Shlomi Fish 的网站 *Better SCM Initiative*（better-scm.berlios.de）介绍了大多数 CVS 的候选替代软件，并对它们的功能进行了系统性的比较（但未标注比较的时间）。

下面从用户的角度快速列举一下最重要的 CVS 命令。修改一个项目的第一步是登录到服务器，检出想要改动的模块。我们在这里要修改的模块叫做 **sort**。

```
$ cvs -d :pserver:username@servername:/path/to/repository login
CVS password: <password>
$ cvs -d :pserver:username@servername:/path/to/repository co sort
```

其中 **pserver** 是联络代码库所用的访问方法，本例中的代码库是一个专用的 CVS 口令服务器。**login** 操作把口令与服务器的口令进行对照检查，并且保留口令的一个副本，供以后的交互使用。**co** 操作直接类似于 RCS 的同名命令。

您现在可以进入 **sort** 目录在本地的副本，开始编辑文件。当您准备把文件再次检入 CVS 库的时候，不需要用**-d** 选项，因为 CVS 在子目录 **sort/CVS** 里留下了一份所有必须的本地信息。

```
$ cd sort
$ vi foo.c
$ cvs commit foo.c -m "Added more efficient sort routine"
```

如果您已经在模块的副本上工作了一段时间，想要用其他人自从您检出该文件后已经对它做的修改来刷新本地副本，可以使用 **cvs update** 命令。**-d** 选项的意思是要包括所有的子目录，而**-P** 选项则要求 CVS 删除任何空目录。

```
$ cvs update -dP
```

要记住，虽然您没有把自己的文件检入到中心代码库，但这也是一种集成。其他用户的改动总有可能与您的改动发生冲突，如果有需要您解决的冲突，CVS 会让您知道。

11.5.5 Subversion：做得好的 CVS

虽然 CVS 是当前占据主导的版本控制系统，但是我们建议系统管理员从 RCS 直接跳到 Subversion。这个软件包由 Karl Fogel 和 Jim Blandy 编写，他们两人在 1995 年创办了 Cyclic Software 公司，销售 CVS 支持合同。到了 2000 年，他们与 CollabNet 签订合同，编写一个代替 CVS 的开放源代码软件。结果就有了 Subversion，经过漫长的孕育阶段，Subversion 的 1.0 版在 2004 年 2 月发布了。

Subversion 具备了上面提到的所有“欠缺”的功能，但是又没有牺牲整洁性和易用性。和在 CVS 里一样，Subversion 系统里有一个集中式的代码库保存所有的版本控制文件。Subversion 能够处理二进制文件，而且速度比 CVS 快。

Subversion 服务器默认是 Apache 2.x 版 Web 服务器中的一个模块。它特别适用于分布式的软件开发工作，但是可能不适合系统管理工作使用。不过还好，Subversion 的开发者提供了第二种以守护进程形式出现的服务器，叫做 **svnserve**。您可以从您的主目录中运行 **svnserve** 来试用 Subversion，但是在正规工作上使用的时候，它应该有自己的用户账号，从 **inetd** 运行。

Subversion 一开始的发布版本使用 Berkeley DB 数据库用于代码库的后端保存，但是 Subversion 1.1

www.TopSage.com

增加了对另一种叫做 FSFS 系统的支持。两种保存方式都各有优缺点。主要区别在于，Berkeley DB 依靠内存映射 I/O 语义，因此不能用 NFS[7]。使用 Berkeley DB 的 Subversion 库必须放在 Subversion 服务器运行的主机上。FSFS 库不受这样的限制。参考 subversion.tigris.org 了解更多优缺点介绍。

设置 Subversion 库很简单。例如，下面几步可以创建一个称为 **admin** 的新 Subversion 库：

```
# cd /home/svn
# mkdir repositories
# cd repositories
# svnadmin create admin
# chmod 700 admin
```

如果您想要自己的 Subversion 库使用 FSFS 格式而不是默认的 Berkeley DB 格式，就要给 **svnadmin create** 命令加上**--fs-type=fsfs** 选项。格式要选得聪明，确定好库的格式之后就难以再变了。

如果您扫一眼 **admin** 目录，就会发现它是一个组织得很好的库结构，其中还有一个 **README** 文件。在子目录 **conf** 里可以找到配置文件 **svnserve.conf**。这个文件告诉服务器守护进程如何提供对新库的访问。下面是一个适用于系统管理文件的配置：

```
[general]
anon-access = none
auth-access = write
password-db = passwd
realm = The Sysadmin Repository
```

因为 Subversion 的设计目标之一就是方便位于不同地点的人们之间的协作，所以它有一个独立于操作系统的访问控制模型。（在同一目录下的）**passwd** 文件里有用户名及其明文（！）口令。这种明文口令着实不好，但作为补偿，口令也绝不会通过网络传送。用户也不会把它们从内存敲出来，所以您也可以把口令设得足够长和随机，以此保证安全性。例如：

```
[users]
tobi = lkadslfkjasdljkhe8938uhau7623rhkdfndf
evi = 09uqalkhlkasdgfprghkjhsdfjj83yyouhfuhe
fritz = kd939hjahkjaj3hkuyasdfaadfk3ijdkjhf
```

passwd 文件的权限自然应该严格设定。

剩下的就是在新库上启动服务器了：

```
# svnserve --daemon --root /home/svn/repositories
```

您作为一个没有特权的用户，现在就可以在网络上的任何地方把 **admin** 模块检出：

```
$ svn checkout --username tobi svn://server.atrust.com/admin checkout

Authentication realm: <svn://server.atrust.com:3690> The Sysadmin Repository
Password for 'tobi': <password>
```

当您第一次输入口令的时候，Subversion 会把它的一个副本保存在您的主目录中创建的**.subversion** 目录里。使用 **svn** 命令可以向项目在本地的副本添加文件，或者从中删除文件：

```
$ cd checkout
$ vi foo.c
$ svn add foo.c
```

[7] 参考 www.sleepycat.com 了解有关这个问题的更多资料。

www.TopSage.com

完成之后，把您的改动提交给 Subversion 库：

```
$ svn commit -m "Initial checkin; added foo.c"
```

不必列出您想要提交的修改过的文件列表，虽然您愿意的话，也可以这么做，**svn** 能分析得到它自己的文件列表。如果省略了**-m** 选项，那么 **svn** 会启动一个编辑器，您可以编辑提交消息。

要从 Subversion 库获得最新更新，可以在项目中运行 **svn update**。和在 CVS 系统里一样，Subversion 在项目的本地副本和主控库里修改过的所有文件上执行一次合并操作。冲突无法解决的文件被标为“conflicted（有冲突）”，您只有先改正这些问题，告诉 Subversion 冲突已经解决了，Subversion 才能让您把文件检入到库中：

```
$ svn resolved foo.c
```

如果您想要知道一个文件里的哪些行是由谁改的，可以要求 Subversion 提供抱怨信息：

```
$ svn blame bar.c
```

这条命令打印出了这个文件包含说明的版本，显示出文件每行内容上次由谁在什么时候修改过（要获得容错性更好，或者说最乐观的特性可以使用 **svn praise** 命令）。Subversion 也能很容易地提供相对于一个特殊日期或者版本的变化信息。例如，如果您想要知道自从 2006 年 6 月 2 日以来，**foo.c** 这个文件发生了哪些变化，就可以用下面的命令：

```
$ svn diff -r "{2006-06-02}" foo.c
```

您可以从 subversion.tigris.org 下载最新版的 Subversion。该软件的标准文档是 O'Reilly 出版的 *Version Control with Subversion* 一书。在 svnbook.red-bean.com 上可以在线获得全文内容。

Subversion 还有一个出色的 Windows GUI；参见 tortoisesvn.tigris.org。我们用它管理本书的源文件。

11.6 本地化和配置

针对您的本地环境对计算机进行调配是系统管理的主要任务之一：告诉系统网络上所有打印机的位置、启动专门发放许可证的守护进程，加上每周清理一次/scratch 目录的 cron 任务，集成对绘图部门的特殊扫描仪的支持等。要注意让这些事情都按有组织和可重复的方式完成，这就是系统结构考虑的中心目标。

牢记下面的要点。

- 用户没有超级用户（root）权限。在正常操作过程中，出现要求获得超级用户权限的现象都很可疑，或许预示着您的本地配置有可疑之处。
- 用户意外地破坏系统。设定内部安全机制，让它保护系统不受意外失误的影响，也不会散布管理权限。
- 应该对行为轻微不端的用户先面谈再惩罚。对于效率不高的管理流程，用户会通过暗中破坏的方式来频繁报复，所以考虑用户不愿遵守纪律的现象可能预示着结构性的问题则是明智之举。
- 以客户为中心。和用户交谈，询问他们什么任务他们觉着最难。找到让这类任务更简单的方法。
- 您个人的偏好只是个人的。让用户有他们自己的偏好。尽可能提供多种选择。
- 当您的系统管理决策影响了用户对系统的使用时，要清楚所做决策的理由。让大家知道您的理由。
- 保持本地文档不断更新，而且容易获得。参考第 30 章的内容了解有关这个话题更多的信息。

www.TopSage.com

TopSage.com

11.6.1 本地化的组织

如果您的站点有 1 000 台计算机，每台计算机都有自己的配置，那么您的大部分工作时间都会花在确定为什么一台机器有个特别的问题，而另一台却没有。显然，解决这种局面的办法就是让每台计算机的配置都一样，不是吗？但是现实世界里的限制，以及用户多变的需求一般不会让您做到这一点。

在系统的可管理性上，多重（*multiple*）配置和无数（*countless*）配置有着很大不同。诀窍在于要把您的设置划分为可以管理的小部分。您会发现有些本地化的部分适用于所有被管主机，而有些部分则只适用于几台主机，而剩下的一些部分专门针对个别机器。

除了从头开始安装系统之外，您还需要持续铺开系统更新。要记住，单台主机有自己在电流、稳定性和开机时间（uptime）方面的要求。

谨慎的系统管理员决不应该一股脑地放出新的软件发布来。相反，软件发布应该按阶段，根据一项逐步的计划提供出来，这一计划会将其他组的需求包含进来，而且还要考虑留些时间，在问题所造成的危害尚且有限时就能发现它们。在您对所要做的改动有一定信心之前决不要贸然更新关键服务器[8]。

不过，您在设计本地化的系统时，要确保把原来所有的数据都保存到了一个版本控制系统中。这个预防措施让您可以跟踪哪些改动已经完全测试过，准备好用于部署。此外，它还能让您辨别出任何引发问题的改动。在这个过程中涉及的人越多，上面考虑到的措施也就越重要。这些观点或许都不正确，但是我们倾向于支持限制软件发布的数量。按您自己的时间安排来升级系统，要比因为外部的一次紧急情况来迫使系统升级好得多。

把基础的 OS 版本和本地化的 OS 版本区分开来是有好处的。根据环境对稳定性的需求不同，您可以使用仅仅修正了 bug 的改动最轻微的本地版本。不过我们发现，一点点增加新特性的过程，比起排着队把多项修改一股脑加到“重大”版本中，冒着服务出现大问题的风险来说，并不会更顺利。

您随时能掌控最多多少“版本”，那就制定多少“版本”，这样做往往是个不错的主意。有些系统管理员认为，软件没有出问题的话，就没理由要去修正。他们指出，无端地去升级系统费时费力，而一切“前卫”在太多情况下都意味着“代价惨痛”。实施这些原则的人就必须乐意接受更多的活动版本类型。

与此相反，喜欢“精悍”的人士们则指出，这些版本自身很复杂不说，要理解（更不用说管理）这些经年累月留下来的各式版本也很困难。他们打出的王牌是安全补丁，一般而言必须打这些补丁，而且要经常打。给过期的操作系统版本打补丁往往不可避免，所以系统管理员就要面临选择，是在有些计算机上跳过升级，还是将这些机器彻头彻尾地升级为一个更新的内部版本。这情形可真算不上好。

上述两种观点没有一种是肯定正确的，但是我们倾向于支持乐意限制版本数量的一方。按照您自己的时间安排做升级，要比遇到外部的一次紧急情况被迫升级更好。

11.6.2 测试

在对外界做出改动之前先测试很重要。至少这意味着您需要测试自己本地的配置变化。不过，您确实也应该测试一下厂商发布的软件。一家主要的 UNIX 厂商曾经有一次发布了一个会执行 **rm -fr ./** 的补丁（在以某种方式施加这个补丁的时候）。先设想一下，没有先测试就在您的机构内到处施加这个补丁会有怎样的情况出现。

如果您使用了像 **apt-get** 或者 Red Hat Network 这样的服务，它们提供自动打补丁的功能，那么测

[8] 安全补丁可能是这一规则的一个例外。重要的是，一旦发现安全漏洞，就要马上补上它。另一方面，安全补丁有时也会引入 bug。

www.TopSage.com

试就成了一件非常重要的事情。关键性的应用系统一定不要直接连到厂商支持的更新服务上。找一台可以牺牲的机器，连到这样的服务上，在测试正确后，再从这台机器把改动传遍站点内的其他机器。在测试过程中禁止更新，否则，在您测试的同时，可能会将更改的顺序弄颠倒，过早地偷偷渗透到您的生产系统之中。

如果您预见到一次有计划的更新活动会引发用户能看到的问题或者变化，那么就要提前通知用户，让他们有机会和您交流一下，他们是否在意您打算做的改动或者做改动的时间。要保证用户有很容易的方法来报告 bug。参考 30.9 节的内容了解有关 bug 跟踪的更多知识。

如果您的单位在地理位置上是分散的，那么就要确保其他办事处也能帮忙做测试。国际性的参与在多语言的环境中特别有价值。例如，如果在美国的办事处里没有人讲日语，那么您最好让东京的办事处来测试任何可能影响到对日文汉字支持的东西。随地点不同而变化的系统参数多得惊人。美国的办事处会测试对采用 A4 纸型打印设施的改动吗？还是非美国的办事处参与才能给大家带来一个惊奇？（译者注：美国普遍采用的纸型为 Letters，而欧洲多采用 A4 纸型。）

11.6.3 本地编译软件

在 UNIX 的早期时代，当时有许多不同的体系结构，程序一般都是以源代码包的形式发布的，通常这种源代码包是**.tar.Z** 文件，您可以解压缩，然后编译它们。一旦构建好了程序，接着就可以把软件安装在像**/usr/local** 这样的地方。现如今，使用软件包管理系统则意味着几乎没有什么程序要以这样的方式来安装。它也意味着系统管理员几乎不需要决定把软件安装到哪儿，因为软件包指出了它们的内容会安装到什么地方。

即使有了简便的软件包管理，可许多人仍然愿意编译他们自己的软件[9]。执行您自己的构建过程能让您对软件的编译选项有更多的控制。它还能让您更偏执，因为您可以检查正在编译的源代码。有些人认为这很重要，但是除非您有时间和技术检查有 20 000 行代码的软件包中的每行代码，我们还是会怀疑增加的安全价值可能非常之低。

因为这个世界上不是每种软件都有用于每种 Linux 的打包版本，所以很可能至少会有几个程序需要您自己安装，特别是如果您的 Linux 主机不是基于 Intel 的 IA-32 体系结构时更是如此。而且，如果您的站点是一个开发站点，您将不得不考虑该把站点自己本地开发的软件安装到什么地方。

从历史上看，本地软件最常用的位置一直是**/usr/local**，这个习惯到了今天仍然被广为遵守，UNIX/Linux 的 FHS（Filesystem Hierarchy Standard，文件系统层次结构标准）规定，在 OS 的初始安装完成之后，要有**/usr/local** 目录，而且为空。许多软件包都要把它们自己安装在那儿。

还有数目不少的其他软件（尤其是商业应用）要求安装在**/usr** 里，在没有软件包管理系统的环境下，这么做一般是个不好的想法[10]。我们的经验是，虽然许多应用软件要安装到**/usr** 下，但把它们安装到别处一般工作起来也不会有问题。对于个别不规矩的软件包，我们建议您把文件实际装在别处，只是在**/usr** 里建一个指向它们的符号链接。另一种方案是检查这个不规矩的应用是不是要读取一个环境变量的值来找到自己的安装目录。这样的特性通常会有文档予以说明，但并不一定如此。

虽然**/usr/local** 是传统位置，但是许多站点发现它成了不好管理的倾倒场。它的传统布局方式（基本上和**/usr** 一样，在**/usr/local/bin** 里放二进制文件，在**/usr/local/man** 里放手册页，依此类推）造成了大量问题：难以安装同一软件的多个版本，目录会变得很大，管理多种体系结构会很痛苦等。

[9] 亲自编译方式的坚定支持者应该看看 Gentoo 这个 Linux 的发行版本，它专为从头重新编译目标系统而设计。

[10] 把随意的软件放到/usr 和其他系统目录之外的关键目的是把本地定制软件同系统软件区分开。在有了软件包管理的情况下，软件包自己就提供了一条审计途径，所以让软件包修改系统目录也更为合理。

www.TopSage.com

11.6.4 发布本地软件

您站点里的本地系统必须做好初始安装和增加更新两方面的工作。更新工作特别富有诀窍。因为您可能想要重复整个本地化过程，只为更新一个文件的权限，所以高效性成了一个关键问题。即使这个过程是自动执行的，“从头再来”的构建模式也会让更新工作成为一个费时费力的过程。

组织本地化工作的一种简单且可扩展的方式就是维护一个树形结构下的文件，这个结构模拟出某个生产机器上的文件系统（框架）。有一个专门的安装脚本可以把这个目录树复制到目标主机上，完成所需的任何编辑工作。

这种类型的设置有几个优点。实现您的本地管理方案需要多少目录树，您就可以维护多少本地化的目录树。有些树是可供选择的，一台机器只能选择其中之一。有些树可以覆盖，能复制到它们之前的树上面。如果必要，本地化的目录树可以覆盖文件，或者也可以完全分解。每个有可能独立安装的树都应该用一个单独的版本控制项目来表示。

覆盖树的方法在实现上有很大灵活性。如果您使用了一个打包系统来发布本地定制信息，那么覆盖树可以简单地汇集起来成为独立的软件包。在软件包中可以包括适当的定制脚本，设为在安装过程中执行。

另一种比较好的实现思路是使用 **rsync** 命令让目标机器和它们的覆盖树保持一致。**rsync** 命令只复制那些过时的文件，所以它对于发布增量变化来说特别高效。单用软件打包系统的方式还很难模拟这种方式。参考 17.3 节了解有关 **rsync** 的更多知识。

11.6.5 解决时间安排上的问题

有些站点定期更新所有的系统。这样的计划可能有点强迫性，但是它也有优势，能够规定客户机的系统过时的时间上限。一些别的站点让系统在启动的时候更新自己。这是一种相对安全的选择，但它也意味着两次更新之间可能会间隔很长时间。有些站点的用户技术水平较高，让用户自己选择什么时候更新自己的机器。这是一种协作性的对用户友好的方案，但是它倾向于需要有另一种方案做补充，因为有些用户从不会自己去更新系统。

更新既可以从更新服务器（通常通过 cron）“推（push）”出，也可以由各个客户机去“拉（pull）”回来。拉回更新的系统赋予客户机更多的控制权，从而能控制自己更新系统的时间安排。例如，一家全球运作的机构可能会发现，采用一个在夜间“拉”回更新的系统更容易实现，在美国的半夜升级在亚洲就成了正午升级。

根据您所管理的机器数量不同以及它们分布的地理范围跨度不同，您可以设置一台软件发布服务器或者按层次组织的多台发布服务器。例如，您可以有一台主控服务器，每次构造软件后向从属服务器发布软件，而从属服务器接下来再直接向客户机发布软件。对于地理上分散的站点来说，这样的部署大大降低了占用的广域网带宽。

11.7 配置管理工具

本地化的系统倾向于“自产自销”。出现这种情况的部分原因在于，所有的站点都不一样，每个站点都有自己的独特之处。不过，NIH（“非此独创”）综合症也是一个重要因素。或许由于缺乏一种占主导地位的开源工具来执行配置管理，使得我们习惯于认为这个问题不在标准工具的受理范围内。

尽管如此，还是有些工具值得您关注一下，就算只是为了让自己清楚为什么不选择采用它们吧。

www.TopSage.com

下面几小节按流行性和相似性的大致顺序列出了比较流行的几种系统。

11.7.1 cfengine：计算机免疫系统

最知名的本地化工具之一是 Mark Burgess 的 **cfengine**。它过去被当作一种“计算机免疫系统”，其操作基于系统应该怎样配置这一模型。当它检测到模型和现实之间存在差异的时候，**cfengine** 会采取适当的步骤让系统遵循模型。因为有这种支撑模型，所以 **cfengine** 对于正在进行的配置管理来说确实有用。

cfengine 可以备份它所修改的文件，为它所做的改动保留一份详细的日志。它还能以一种“无为（no-action）”模式来运行，在这种模式下，它只是说要做更改，但实际又不会做更改。

您可以使用 **cfengine** 自己的特殊语言来描述您想要怎样配置自己的计算机。您可以指定诸如“文件 **xyz** 必须位于**/etc** 下，权限为 644，而且属于 root”这样的规则。您还可以根据单个文件的内容编写规则。例如，您可以指定**/etc/hosts** 文件必须包含“router 192.168.0.1”一行内容。如果这个文件里没有这一行，那么 **cfengine** 就会把它给加上。

cfengine 的配置语言能让您根据主机名、操作系统或者子网等因素启动各规则。这项功能可以更容易地实现编写一个配置文件，却能覆盖管理域内所有主机的需求。

下面是 UNIX 环境下的一个简单例子。它保证在 Sun 主机上的**/bin** 是一个链接到**/usr/bin** 的符号链接，对老式的 OSF 机器还要检查其他一些链接，并且删除**/var/scratch** 目录下所有 7 天前的文件。

```
control:
actionsequence = ( links tidy )
links:
sun4::
/bin -> /usr/bin
# other links
osf::
# some osf specific links
tidy:
/var/scratch pattern=* age=7 recurse=inf
```

参考 **cfengine** 的主页 www.iu.hio.no/cfengine 了解更多信息。

11.7.2 LCFG：大规模配置系统

LCFG 最初于 1993 年由爱丁堡大学的 Paul Anderson 开发。它最近的形式称为 LCFG(ng)，并且赢得了若干大学校园之外的用户。LCFG 主要面向大型 Red Hat 安装的管理，但是正在向 Mac OS X 和 Solaris 上移植。LCFG 的网站是 www.lcfg.org。

和 **cfengine** 类似，LCFG 也定义了一种专门的配置语言。所有被管机器的配置都保存在一台中央服务器上的一组主控配置文件里。LCFG 从这些文件生成定制的 XML 文件，它们描述了每台被管主机的配置。在中央服务器上的守护进程监视主控配置文件的变化，根据需要重新生成 XML 文件。

XML 文件在一台内部的 Web 服务器上公布出来，客户机随后就可以从这台服务器拉回它们自己的配置。客户机使用各种组件脚本来按照 XML 的规定配置它们自己。

11.7.3 Arusha 项目（ARK）

Arusha 这个项目由格拉斯哥大学的 Will Parain 建立。Arusha 背后的动机是大多数系统管理员做的事情本质上都相同，但不是完全相同。按 Will 自己的说法，“我想让我的服务和他们的服务一样，但

www.TopSage.com

稍有不同……”位于 ark.sf.net 的 Arusha 网站详细介绍了该系统的哲学基础。

Arusha 基于一种叫做 ARK 的面向对象的语言。它用 XML 来表示，让系统管理员定义诸如主机、许可证、网站、邮递列表等这样的对象。您可以用 ARK 描述这些对象之间的依赖关系，还可以提供澄清它们之间相互关系的方法和数据。和面向对象的软件开发一样，可以重用对象；也可以有子类，只重载必须要改变的部分，使之能在您的本地环境里起作用。

11.7.4 Template Tree 2：cfengine 的帮手

Template Tree 2 由瑞士苏黎世理工（Swiss Federal Institute of Technology）的 Tobias Oetiker 编写。它是一种基于组件的由中央配置驱动的系统。它通过采用一种两层的方法来定义一个站点的配置，以此降低复杂性，并且能够用于重新定位无盘计算机的根目录。

在下层，系统由若干“特性集”组成。一个特性集就是一组文件加一个 META 文件的集合，后者描述了这些文件必须怎样安装到目标系统上去。而特性可以是从网络配置到最新版 OpenSSH 的任何东西。特性可以接受可配置的参数，这些参数能够在主控配置文件中设置。

上层的配置是一个站点的主控配置文件，您可以把这些特性放在一起，把它们同若干计算机或者若干个计算机组联系起来。在这一层上，您必须指定每个特性的配置参数。例如，邮件服务器特性的参数之一是邮件的域名。

Template Tree 2 把来自主控配置文件的信息和各个特性的 META 文件结合起来，生成整个站点的 **cfengine** 配置文件。因为每个特性必须包含有关其目的和用法的文档，所以 Template Tree 2 也能产生组成的说明文档。

11.7.5 DMTF/CIM：公共信息模型

分布式管理任务组（DMTF，Distributed Management Task Force）联合了 3 000 多家积极参与者，从 1992 年起开始开发自己的公共信息模型（CIM，Common Information Model），目标是为一个面向对象的跨平台管理系统建立标准。

按照 DMTF 自己的说法，CIM 是“一种管理模式（schema），既在分类和关联方面，又在基础类方面（用于根据对被管环境的描述建立一个通用框架），从基础拓扑的层次上建立一个通用的概念框架。”或者不管是什么都行。

从微软到 Sun 的所有大公司都是 DMTF 的成员。遗憾的是，该组织所制定的标准却展示出用晦涩和时髦辞藻舞文弄墨的做法，令人印象深刻。参与的公司似乎都热衷于表现其对于标准化的积极愿望，而对标准的内容则无所谓。标准的核心在于 XML 和面向对象。然而，我们尚未看到一种建立在这些标准上的有意义的产品。

因为标准化的工作正在由非开源的老商业公司主推，因此结果不难预料。例如，Sun 公司出品的一种管理无盘客户机的系统，就是在 Solaris 8 上基于这些技术实现的。凑近再看就会发现，其实是一个基于 Java 的服务器被一个基于 Java 的客户机触发，然后前者提供一个 shell 脚本，这个脚本再完成实际工作。当出现问题的时候，该系统的手册指导用户去检查日志文件，它是个二进制结构的文件，只有通过一个基于 Java 的日志查看程序才能访问。

如果这种困境还算有优势的话，那就是 DMTF 的努力至少要求厂商按照一种开放标准，提供能以程序方式访问其系统的接口。对于 UNIX 和 Linux 环境而言，这并不算新鲜，但 DMTF 却不是一个 UNIX 的组织。它包括 Cisco、Microsoft、Symantec 和许多别的公司，这些公司历史上几乎没有提供使其系统脚本化的有效途径。尽管目前还很缺乏标准的实现，但对于这些产品，有个配置 API 也是好事情。

www.TopSage.com

大家网 TopSage.com

11.8 通过 NFS 共享软件

额外的软件应该装在哪儿？装在各个客户机上，还是装在中央文件服务器上，然后通过 NFS 共享？Linux 的标准回答是“装在客户机上”，但采用 NFS 的方案却能让更新的速度更快（更新 10 台 NFS 服务器要比更新 1 000 台客户机速度更快，也更可靠），而且节省了客户机上的硬盘空间（在硬盘容量高达 400GB 的世界里这不算多大问题）。

问题实际上归结到了可管理性和可靠性的比较上。基于网络文件系统的访问方法是集中式的，日常管理起来更容易，能立即在所有的客户机上修改 bug 和安装新软件包。不过，通过网络运行要比访问本地硬盘速度慢一些。此外，网络服务器模型增加了对网络和文件服务器的依赖性，不仅因为它增加了潜在的故障点，而且因为它要求客户机和服务器必须就某些方面取得一致，比如要用哪些共享库，这些共享库安装的版本等。底线在于，NFS 软件库是一种高级管理技术，只应该用在允许高度集中协作的环境中。

如果您工作的环境里有高性能的网络，能够买得起用于中央服务器的快速 RAID 盘阵，那么您会发现网络服务器实际的性能要比本地 IDE 硬盘快得多。无论性能好坏，性能的影响可能都不大，对于性能的考虑不应该在体系结构设计上占据主导地位。

一般来说，异构系统的网络从共享软件库获得的好处最大。如果您的站点按标准只采用一种操作系统，而这种操作系统又提供了不错的软件包管理设施，可能最好还是坚持采用本来的系统，不用网络系统。

11.8.1 软件包的名字空间

传统的 UNIX 把新软件包的内容分散放到多个目录下。库放在**/usr/lib** 下，二进制程序放在**/usr/bin** 下，而文档放在**/usr/share/docs** 下等。尽管 FHS（Filesystem Hierarchy Standard，文件系统层次结构标准）有助于让文件的位置多一点能预测的可能性，但 Linux 还是或多或少继承了与 UNIX 一样的体系。

这一约定的优点是文件都会出现在已知的地方——只要您的环境变量 PATH 指向了**/usr/bin** 和其他放二进制程序的标准目录，那么马上就能用上新安装的程序。

而它的缺点在于，必须能清楚地确定文件的来源（借助软件包管理系统），而且分散的文件难以在网络上共享。还好，乐意多做些额外工作的系统管理员有个不错的办法能摆脱这个问题：软件包的名字空间。

这个方案的要点是把每个软件包安装到它自己单独的根目录中。例如，您可以把 **gimp** 安装到**/tools/graphics/gimp** 里，二进制程序的位置在**/tools/graphics/gimp/bin/gimp**。接下来，您就可以通过在目录（比如**/tools/bin**）里放上符号链接，为您自己的工具集重新建立一个集中的二进制目录：

```
/tools/bin/gimp -> /tools/graphics/gimp/bin/gimp
```

随后，用户可以把**/tools/bin** 加到他们自己的 PATH 变量里，好能找到所有的共享工具。

目录**/tools** 的结构可以有多种选择。分层组织的方法（例如，**/tools/graphics**、**/tools/editors** 等）既方便浏览，又加快了性能。您可以在自己的命名约定中包含软件版本、硬件体系结构、操作系统或者负责人姓名的首字母，让同一组工具集能服务多种客户机。例如，Solaris 用户可以在他们的 PATH 中包括**/tools/sun4/bin**，而 Fedora 的用户则用**/tools/fedora/bin**。

当您安装了一个重要工具的新版本之后，不限期地保留它的老版本是个好想法，特别对于用户已经在使用该工具的项目上投入大量时间和精力的情况更要这么做。理想情况是，工具的新版本要向后

www.TopSage.com

兼容旧的数据文件和软件，但在实际操作时，出问题的情况却司空见惯。好的做法是，要求用户克服一些配置上的麻烦，仍能用上一个软件包的较早版本；不好的做法是，打断用户现在的工作，让他们自己去处理后果。

11.8.2 依赖关系的管理

有些软件包要依赖若干库或者其他软件包。当您通过一个软件包管理系统在本地安装软件的时候，您可以获得很多辅助手段，解决好这些问题。不过，当您建立自己站点用的网络软件库的时候，您就必须明确地处理这些问题。

如果您按照和管理应用软件一样的方式来管理库，那么可以编译您的工具，让它们使用被共享的 **/tools** 目录下的库。这样的做法可以让您同时使用一个库的多个版本。因为依赖库的应用软件链接到了库的特定版本上，所以即便发布了这个库的新版本，这种设置仍然保持稳定。缺点是随着时间的推移，对于使用和维护来说，这种设置会变得相当复杂。

要经得住诱惑，不要在全局的**/tools/lib** 目录下，用不带版本号的通称给链到共享库的链接起名字。如果您改变了链接，就会遇到不确定且难以诊断的问题。共享库系统就是为解决一些这样的潜在问题而设计的，但是要在一个复杂的设置下安全地使用它才有意义。

要让编译时的链接器使用一个共享库的特定版本，所需的确切步骤随系统的不同而不同。在 Linux 下，您可以设置环境变量 LD_LIBRARY_PATH 或者使用链接器的-R 选项。

11.8.3 封装脚本

遗憾的是，做到库一级的兼容只解决了问题的一半。有的工具还要调用别的工具，这种情况直接造成了另一个发生冲突的机会。例如，假定有个叫做 **foo** 的工具程序需要频繁调用名为 **bar** 的工具。如果您更新了 **bar** 的默认版本，那么您可能会发现 **foo** 突然就不干活了。在这种情况下，您可以断定 **foo** 依赖 **bar** 不再支持的某种行为（或者至少不再是默认的行为）。

如果您的软件库支持多版本（例如，**/tools/util/bar-1.0** 和**/tools/util/bar-2.0**），就可以通过把 **foo** 原来的版本改为 **foo.real**，然后再用一个小的封装脚本来代替它：

```
#!/bin/sh
# make sure the program finds any files co-packaged with it
# first even if it does not use an explicit path.
PATH=/tools/util/bar-1.0/bin:$PATH
export PATH
exec /tools/util/foo-1.0/bin/foo.real "$@"
```

现在可以用一个定制过的 PATH 环境变量来启动 **foo**，它会优先调用 **bar** 的老版本而不是新版本。

封装程序（wrapper）是一种功能很强的工具，它能够解决的不仅仅是软件包依赖性的问题，而且还能解决像安全、体系结构或者操作系统依赖性、以及跟踪使用情况等这样的问题。有些站点会把所有的共享库包裹起来。

11.8.4 实现工具

在系统管理工作中，维护共享软件库是一个共同面临的问题，系统管理员已经开发出了各种软件方案，从而有助于让这个过程更容易进行。和本地化一样，“自产自销”的方法似乎占据了主流，但总还是要找找看，在能获得的系统里是否会有满足需要的。

要找到其中一些经典系统，可以用 google 搜索一下 CMU Depot、CERN ASIS 或者 CITES Encap。

www.TopSage.com

这些项目中有些开发还很活跃，而另外一些则已经被放弃了，因为受到了来自 Windows 主机，以及广泛采用 RPM 软件包更新的压力。

但是仍然还有几个比较新的项目。

- GNU Stow（www.gnu.org/software/stow）在一个保存二进制程序的中心目录里建立很多链接，链接到软件包所在目录里的实际二进制程序上。
- Chrisophe Kalt 开发的 Swpkg（web.taranis.org/swpkg）是一套工具，能够完成构建一个共享软件库所需要大部分步骤。
- Peter Kristensen 的打包管理计划（Pack Management Project）则一开始就雄心勃勃，不仅要提供一个安装软件的系统，而且要能重新对软件打包（着重在 Solaris 上）。重新打包的这部分目标已经被放弃了，但是在软件工具上的开发工作仍然在继续进行。
- 在瑞士苏黎世理工（Swiss Federal Institute of Technology）里创建的 SEPP 系统（www.sepp.ee.ethz.ch），它实现了前面讨论的大部分想法。

11.9 推荐软件

有些工具非常重要，我们认为必须要在所有的机器上都安装。我们喜欢的其他一些软件包则可能只对选出的一些机器来说是必要的或者有用的。不过，考虑到现如今硬盘空间的价格，到处都安装所有的系统软件，维护本地软件树的一致性，这样做比较容易。表 11.1 给出了我们认为必须安装的软件清单。

表 11.1 必备的软件包

软 件 包	说明和建议
ssh/scp	Secure Shell（安全 shell）——使用加密技术，不会泄漏口令
sudo	**su** 的替代程序——增加了控制和日志功能
sendmail 或者 **postfix**	可以选用的邮件传输程序
traceroute	网络路由追踪——调试网络问题时使用
tcpdump 或者 Wireshark	可以选用的网络嗅探器——用于分析网络流量
gzip	GNU 的数据压缩程序——解压缩下载文件时需要
mozilla 或者 **firefox**	可以选用的 Web 浏览器——必须要有
RCS/CVS/Subversion	版本控制系统——对于系统管理员和用户都很有用
Perl	脚本化的语言——解决问题的通用工具

表 11.2 给出了我们认为“不错但不必要”的程序清单，我们把这张表分成“系统管理员”和“普通用户”两部分。

表 11.1 和表 11.2 中列出的所有程序都是自由软件。其中大部分软件都能从 Web 上下载（可以试试 freshmeat.net），但是有几个仍然只通过 FTP 来发布。如果您的发行版本中没有包含它们，那么就要用搜索引擎从网上找到它们，或者看看本书的其他地方有没有给出下载链接（检查索引看看在别的地方有没有讨论过这个命令）。

表 11.2 有用的软件包

	软 件 包	说明和建议
系统管理员	**gcc**	C/C++编译器——GNU 的高质量编译器
	BIND	名字服务工具——取得当前的版本（出于安全考虑）[a]
	npasswd	**passwd** 的替代程序——强迫用户选择安全性好的口令

www.TopSage.com

续表

	软 件 包	说明和建议
系统管理员	**xntpd**	时间守护进程——保持机器的时钟准确和同步
	Samba	Windows SMB——与 Windows 系统共享文件/打印机
	Apache	Web 服务器
	Squid	Web 代理和缓存软件
	imapd/procmail	邮件工具——访问和过滤电子邮件
	mrtg、RRDtool	监视工具——监测网络流量和其他数据
用户	Acrobat Reader	PDF 文件显示，Adobe 公司的一种（免费）好工具[b]
	gimp	位图编辑器
	xfig	简单的 X 窗口绘图程序
	GnuPG	签署、验证和加密消息
	nvi/vim	类似 **vi** 的文本编辑器——建议系统管理员使用
	emacs	文本编辑器/操作系统——高级用户的好工具
	pico、**nedit**	文本编辑器——入门用户的好选择
	enscript、**mpage**	打印工具程序——很好的打印机，在单页纸上打印多张的制版程序
	pine	适合入门用户的邮件阅读程序
	thunderbird	喜欢 GUI 的用户的邮件阅读程序
	mh/exmh	处理大量邮件的邮件阅读程序（这个程序得到的维护不太好）
	glimpse	索引工具——对文件做索引，执行快速查找
	gs、**gv**、**ghostview**	预览和打印 Postscript 文档的工具

a. BIND 软件包包括 dig.host 和 nslookup。
b. 自由软件 Evince 是 Acrobat Reader 在 GNOME 下的一种合理替代软件。

11.10 推荐读物

INTEL CORPORATION AND SYSTEMSOFT. *Preboot Execution Environment (PXE) Specification, Version 2.1*,1999. www.pix.net/software/pxeboot/archive/pxespec.pdf

MINTHA, JIM, AND PIETER KRUL. *UltraLinux FAQ*. www.ultralinux.org/faq.html

PXELinux Questions. syslinux.zytor.com/pxe.php

RODIN, JOSIP. *Debian New Maintainers' Guide*. www.debian.org/doc/maint-guide. 这篇文档里有关于.deb 软件包的知识。

SILVA, GUSTAVO NORONHA. *APT HOWTO*. www.debian.org/doc/manuals/apt-howto

HOHNDEL, DIRK, AND FABIAN HERSCHEL. *Installation of Linux Systems Using YaST*. www.usenix.org/events/lisa99/full_papers/hohndel/hohndel_html

STÜCKELBERG, MARC VUILLEUMIER, AND DAVID CLERC. *Linux Remote-Boot mini-HOWTO: Configuring Remote-Boot Workstations with Linux, DOS, Windows 95/98 and Windows NT*. 1999. tldp.org/HOWTO/Remote-Boot.html

The Red Hat Enterprise Linux System Administration Guide. www.redhat.com/docs

WACHSMANN, ALF. *How to Install Red Hat Linux via PXE and Kickstart*. www.stanford.edu/

www.TopSage.com

~alfw/PXE-Kickstart/PXE-Kickstart.html

BURGESS, MARK. *Cfengine: A Site Configuration Engine*. USENIX Computing Systems, Vol 8, No 3. 1995. www.cfengine.org

OETIKER, TOBIAS. *SEPP: Software Installation and Sharing System*. Boston: LISA 1998. people.ee.ethz.ch/oetiker/sepp

11.11 习题

E11.1 列出 Kickstart 和 Auto YaST 之间的不同之处。您会在什么时候选择不用发行版本特有的安装程序？

E11.2 从 subversion.tigris.org 取得并安装一份 Subversion 软件。配置好 svnserve 并且创建一个库。要从本地网络的任何地方都能使用这个 Subversion 库，但仍然保持好安全性，该怎么做？

E11.3 复查本地软件在您站点上的组织方式。系统能扩展吗？它易于使用吗？展开讨论。

★E11.4 列出创建一个 RPM 软件包所需的步骤。使用这个流程来将您在其他计算机课上编写的软件打成 RPM 软件包。

★E11.5 重复 E11.4，但这次是创建一个 Debian 格式的.deb 软件包。

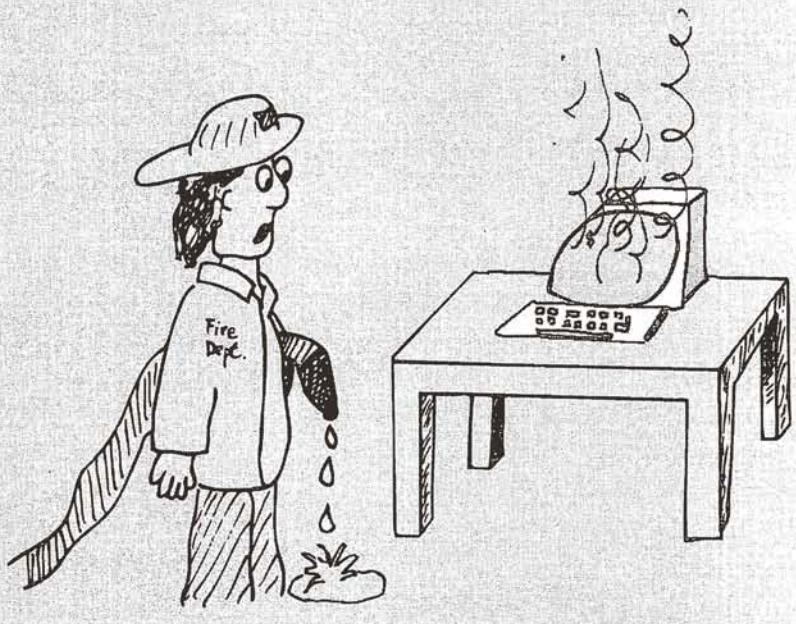
★★E11.6 配置好您所选择的网络安装程序，用您的服务器安装一台新机器。列出执行此项任务所需的所有步骤。其中的难点是什么？您采用自己选择的安装程序时发现有什么扩展性的问题吗？

www.TopSage.com

大家网
TopSage.com

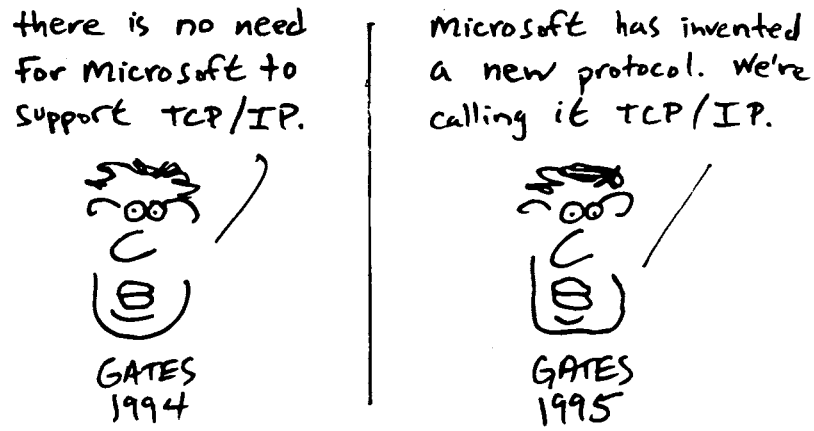
第二部分

网络管理技术



www.TopSage.com

第 12 章　TCP/IP 网络



网络在现代计算技术中的重要性怎么说都不为过，然而人们似乎并没有因此而停止试图夸大网络作用的行为。如今在很多地方，计算机的主要用途是访问 Web 和使用电子邮件。截止到 2006 年初，Internet 用户估计超过 10 亿，比 2001 年的估计数字增长了 55%。局域网、Internet 连接、Web 站点以及网络相关软件的维护是大多数系统管理员的基本工作内容。

TCP/IP 是 Linux/UNIX、Mac OS、Windows 以及其他大多数操作系统最常用的连网协议族。它也是 Internet 的母语。TCP 代表传输控制协议（Transmission Control Protocol），而 IP 指的是网际协议（Internet Protocol）。

尽管使用 TCP/IP 的各种设备有许多差异，但是它们却能够交换数据（“互操作”）。作为该协议族中下层传输协议的 IP 是 Internet 的基础。TCP 和 UDP（User Datagram Protocol，用户数据报协议）是建立在 IP 的基础上、向特定的应用程序传送分组（packet）的传输协议。

TCP 是一种面向连接的协议，它可以使两个程序之间很容易进行对话。它在很大程度上与电话类似：您所说的话被传送给您拨叫的那个人，而对方的话也会回传给您。甚至当双方都没有讲话的时候，也会保持这种连接。TCP 提供了可靠的数据传送、流控制和拥塞控制。

UDP 是一种面向分组（packet）的服务，它和通过邮局发信的原理类似。UDP 是一种无连接协议，它不具有任何形式的拥塞控制，也不保证分组的接收顺序和发送顺序相同（甚至都不能保证分组能被发送到目的地）。

TCP 是一种很文雅的协议，它会迫使互相竞争的用户共享带宽，并且在一般情况下按照有利于网络整体效率的方式工作。另一方面，UDP 则会尽可能快地处理分组。

www.TopSage.com

随着 Internet 的日益普及和拥挤，我们需要让流量最大限度地采用 TCP 以避免拥塞，并且有效地共享可用带宽。现如今，TCP 协议占据了 Internet 流量的绝大部分，而 UDP 和 ICMP 分别占 Internet 流量的第二和第三位，但是和 TCP 的差距非常大。虽然像游戏、音乐、语音以及视频这类应用都是 UDP 应用，但远不如 Web 应用，以及像 BitTorrent 这样的流行应用来得普遍，后者很占带宽，但却用 TCP 而不是用 UDP 来进行传输。

本章从 Internet 的技术方面介绍了 TCP/IP 协议。遗憾的是，就连最基本的连网知识都是一个很大的专题，难以在一章里讲全。本书与网络相关的其他章还有第 13 章（介绍路由技术）、第 19 章（介绍网络管理和故障排除）、第 15 章（介绍域名系统）和第 20 章（介绍网络安全）。

下面几节的内容包括 Internet 的协议与政治的背景知识，它们结构松散，而且观点性较强。读者可以直接跳到 12.4 节学习 IP 的详细内容，也可以跳到 12.8 节了解特定发行版本里的网络配置信息。

12.1 TCP/IP 和 Internet

TCP/IP 和 Internet 的共同历史可以追溯到几十年以前。Internet 技术的成功在很大程度上要归功于 TCP/IP 良好而又灵活的设计，以及它是开放和非专有协议族这一事实。反过来，Internet 的影响力也帮助 TCP/IP 战胜了其他几种与之竞争的协议族，它们因为政治或商业因素而曾经一度流行过。

12.1.1 简要的历史回顾

与普遍的认识相反，Internet 既不是微软公司 1995 年出台的产品，也不是某位前美国副总统的创造物。现代 Internet 的始祖是一个叫做 ARPANET 的网络，它是美国国防部的研发部门 DARPA（Defense Advanced Research Project Agency，国防高级研究计划署）在 1969 年建成的。ARPANET 最终演变成了连接超级计算机站点和区域性网络的 NSFNET 主干网。

到了 20 世纪 80 年代末，网络已经不再是一个研究项目，美国国家科学基金会把它自己从网络连接业务中抽身出来的时机已到。经过几年时间，Internet 过渡成为商业性的网络，而 NSFNET 于 1994 年 4 月关闭。今天的 Internet 主干网是属于网络服务提供商（Internet service provider，ISP）的私有网络的一个集合，这些网络在许多称为对等点（peering point）的地方互连起来。

在 20 世纪 80 年代中期，Internet 基本上由原 ARPANET 的网点和少数大学里的 DEC VAX（Digital Equipment Corporation，DEC）计算机所组成，这些计算机运行 Berkeley UNIX，并处于通过 56kbit/s 的租用数字电话线连接起来的 10Mbit/s 的以太网之中。每当 9 月份学生们返校的时候，Internet 就会发生拥塞而崩溃。这个时候 Van Jacobson（劳伦斯伯克利实验室网络研究小组中的一位研究员）就要检查协议在重负载下的运行状况并对它们进行修改。我们今天知道的 *slow start*（慢启动）、*congestion avoidance*（避免拥塞）、*fast retransmit*（快速重发）以及 *fast recovery*（快速恢复）等算法都是在这个环境下诞生的。

摩尔定律（即硬件的速度每 18 个月增长一倍这一定律）和市场压力极大地加速了 Internet 的发展。从 20 世纪 80 年代末期 TCP 算法已经稳定下来的时候算起，网络接口的速度提高了 1 000 倍（从早期的 10Mbit/s 以太网上 6%的效率，到万兆以太网上接近 100%的效率），租用电路的速度提高了 12 000 倍，而主机的总数增加了 80 000 倍。

我们仍然健在而且富有活力的 Internet 现在运行的基本上仍是 30 年前为一种极为不同的 Internet 所设计的 TCP/IP 协议族，只有曾经设计过某种软件系统，并且目睹它被下一代硬件或下一版本的操作系统所废弃的设计者们才懂得，这是多么令人震惊的事实。我们向 Bob Kahn、Vint Cerf、Jon Postel、Van Jacobson 以及 Internet 的其他所有创造者致敬。

www.TopSage.com

12.1.2 当今 Internet 的管理方法

Internet 的发展始终都是一个协作而开放的过程。由于 Internet 已经成为了世界经济的一种驱动力量，所以各方面都担心 Internet 会掌握在一群计算机“迷”的手中，从而可能接受一些美国政府的指示。不管人们愿意与否，Internet 的管理机构已经出现了。

有关的几家机构如下。

- ICANN（Internet Corporation for Assigned Names and Numbers，Internet 名称和编号分配组织）：如果说存在着主管 Internet 的机构，那么就是这个机构了（www.icann.org）。
- ISOC（Internet Society，Internet 社团）：ISOC 是代表 Internet 用户的一个会员组织（www.isoc.org）。
- IETF（Internet Engineering Task Force，Internet 工程任务组）：这个组织监督 Internet 技术方面的开发和标准化工作。它是一个开放的论坛，任何人都可以参加（www.ietf.org）。

在这些机构中，ICANN 的工作任务最重：将其自身建设成为管理 Internet 的权威机构，消除以前所犯的错误以及预计 Internet 未来的发展。

除了这些组织和机构之外，学术研究人员的国际组织、政府机构以及产业界巨头组成了网络技术协会，称为 Internet2。这些组织群策群力，贡献出推动 Internet 前进的关键思想和技术。与普通媒体报道的内容正相反，Internet2 不是独立于 Internet 的另一个网络。虽然它使用了私有主干网络 Abilene 做连网技术方面的研究，但是它并不会取代我们今天所熟知的 Internet。从 www.internet2.edu 可以了解到更多有关 Internet2 的知识。

12.1.3 网络标准和文献

Internet 界的技术活动都汇总在叫做 RFC 的文档里：RFC 即 Request for Comments（请求评议）。协议标准、修改提议、以及信息公告通常最终都归结为 RFC。RFC 一开始作为 Internet 草案（Draft），然后人们通过大量的电子邮件和 IETF 的会议对此开展辩论，最后它们不是被废弃，就是提升编入 RFC 系列之中。它鼓励对一项 RFC 草案或者提议有意见和建议的任何人给出自己的评议。除了负责 Internet 协议的标准化之外，RFC 有时候仅仅只是现有实际情况的说明文档或者对情况的解释。

RFC 是顺序编号的，现在大约有 4 000 个 RFC。RFC 还具有描述性的标题（如 *Algorithms for Synchronizing Network Clocks*，同步网络时钟的算法），但为了避免混淆，通常都用号码引用它们。RFC 的内容一经发表就不会再改变了。对 RFC 的更新以新 RFC 的形式颁布，并有它自己的引用号。按照惯例，更新过的 RFC 包括所有要保留的适当内容，所以新的 RFC 可以完全代替旧的 RFC，至少理论上如此。

发表 RFC 的过程本身也在标题为 *Internet Official Protocol Standards*（Internet 正式协议标准）的 RFC 里有说明文档。这个 RFC 还包括目前针对各种不同协议标准的大部分 RFC 的索引。由于这方面的信息变化频繁，所以每出现 100 个新的 RFC，这个 RFC 就重新发表一次。Internet 标准处理过程本身则在 RFC2026 里详细说明。另一个有用的基本 RFC 是 RFC2555，RFC 的 30 年历史（*30 Years of RFCs*），这个 RFC 说明了 RFC 系统背后的一些文化和技术背景。

不要被 RFC 里所看到的大量技术细节所吓跑。大多数 RFC 的内容都是简介、总结以及基本原理，这些内容对于系统管理员来说是很有用的。有些 RFC 专门写成概述或者一般性介绍。虽然 RFC 可能算不上学习某个专题的最佳方式，但它们权威、简洁、而且免费。

并不是所有的 RFC 都是干巴巴地充满了枯燥无味的技术细节。我们最喜欢的一些有意思的 RFC（往往写于 4 月 1 号）有 RFC1118、1149、1925、2324 和 2795：

- RFC1118——*The Hitchhiker's Guide to the Internet*；

www.TopSage.com

- RFC1149——*A Standard for the Transmission of IP Datagrams on Avian Carriers*[1]；
- RFC1925——*The Twelve Networking Truths*；
- RFC2324——*Hyper Text Coffee Pot Control Protocol（HTCPCP/1.0）*；
- RFC2795——*The Infinite Monkey Protocol Suite（IMPS）*。

这些 RFC 都是很好的读物，并且略微深入地介绍了设计和建立我们的 Internet 的人物。

除了已经分配的序列号之外，还可以为 RFC 分配 FYI（For Your Information，供您参考）号、BCP（Best Current Practice，当前最佳做法）号或者 STD（Standard，标准）号。FYI、STD 和 BCP 是具有特殊内容或重要性文档的 RFC 子系列。

FYI 是面向广大用户的介绍性或信息类文档。通常，它们是研究某个不熟悉的领域的一个良好起点。STD 记载了已经完成了 IETF 评审和测试过程，并被正式采纳为标准的 Internet 协议。BCP 文档记载了向 Internet 站点推荐的步骤，它们由管理性建议所组成，对于系统管理员而言，BCP 经常是最有价值的 RFC 子系列。

RFC、FYI、STD 和 BCP 在它们各自的系列里按顺序编号，所以一个文档可以具有几种不同的标识号码。例如，RFC1635，*How to Use Anonymous FTP* 也是 FYI0024。

可以从许多来源获得 RFC。在 RFC 相关事务分送中心 www.rfc-editor.org 上有一份保存 RFC 的活动镜像站点的清单。

12.2 连网技术概述

既然我们已经给出了一些背景知识，现在就让我们看看 TCP/IP 本身。TCP/IP 是一个“协议族”，即设计成平稳协作的一组网络协议。它包括若干组成部分，每个部分都由一个标准 RFC 或者一系列 RFC 定义。

- IP（Internet Protocol，网际协议）把数据分组从一台计算机路由至另一台计算机的协议（RFC791）。
- ICMP（Internet Control Message Protocol，网际控制报文协议），提供几种对 IP 的低级支持，其中包括错误消息、路由支持以及调试帮助（RFC792）。
- ARP（Address Resolution Protocol，地址解析协议）将 IP 地址翻译为硬件地址（RFC823）[2]。
- UDP（User Datagram Protocol，用户数据报协议）和 TCP（Transmission Control Protocol，传输控制协议）将数据传递给目标计算机上的特定应用程序。UDP 为各个报文提供了不作验证的“尽力而为（best effort）”传输，而 TCP 可以保证在两个主机的进程之间的可靠、全双工、有流控、可纠错的会话（RFC768 和 793）。

TCP/IP 是围绕表 12.1 所示的分层方案而设计的。

表 12.1 TCP/IP 网络模型

层	功 能
应用层	终端用户应用程序
传输层	把数据传送至应用[a]
网络层	基本的通信、寻址和路由功能

[1] BLUG（挪威卑尔根 Linux 用户组，Bergen Linux User Group）的一群 Linux 爱好者实际上实现了 RFC1149 中定义的 Carrier Pigeon Internet Protocol（信鸽承载 IP 协议，CPIP）。参见 www.blug.linux.no/rfc1149，以获得更多信息。

[2] 这实际上是一个小小的谎言。ARP 实际上并不是 TCP/IP 的组成部分，它可以在其他协议族中使用。但它是 TCP/IP 在大多数 LAN 介质上运行办法的组成部分。

www.TopSage.com

续表

层	功 能
链路层	网络硬件和设备驱动程序
物理层	电缆或者物理介质本身

a. 可以用于处理可靠性和流控制问题。

在 TCP/IP 得以实现和部署之后，国际标准化组织也推出了它自己的 7 层协议族，这个 7 层协议族叫做 OSI。它是一个由专家们提出的完美而累赘的设计，由于它的复杂性和效率低下，所以从未得到实际采用。一些人认为应该在最初的 7 个 OSI 层上面再加一个经济层和一个政治层[3]。

图 12.1 显示了 TCP/IP 的各种组件和客户端在一般结构和分层方案中的布局。

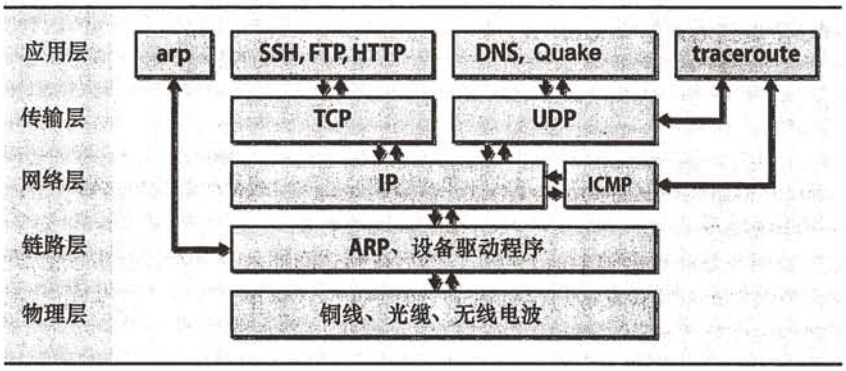


图 12.1 TCP/IP 的快乐大家庭

12.3 分组和封装

UNIX Linux 可以支持各种物理网络，包括以太网、FDDI、令牌环、ATM（Asynchronous Transfer Mode，异步传输模式）、无线以太网以及基于串行线的系统。硬件是在 TCP/IP 体系结构中的链路层内管理的，高层的协议并不知道或者不关心所使用的特定硬件。

数据以分组（或者称为包，*packets*）的形式在网络上传输，突发数据脉冲的最大长度由链路层决定，每个包都由包头（header）和有效载荷（payload）组成。包头说明包来自何方，去向何处。它还可以包括检验和、特定于协议的信息或者其他处理指令。有效载荷指的是要传送的数据。

原始的数据单位名称依协议层而定。在链路层上，它叫做帧（*frame*），在 IP 层上叫做分组（*packet*），在 TCP 层上叫做片段（*segment*）。在这里，我们用“包（*packet*）”作为所有这些情况下的通用术语。

在一个包沿着协议栈向下传送（从 TCP 或者 UDP 传输到 IP，从 IP 到以太网，再从以太网到物理导线）并准备发送的时候，每个协议都会加上它自己的头部信息。各个协议最后得到的包都会成为下一个协议所生成包的有效载荷部分。这种嵌套叫做封装。在接收端的主机上，封装在包沿着协议栈向上回传时解封。

例如，通过以太网传输的一个 UDP 包就包括 3 个不同的包装或者说信封。在以太网导线上，它用一个简单的帧头封装成帧，这个帧头列出了源和下一跳目的地的硬件地址、帧的长度以及帧的校验和（CRC）。以太帧的有效载荷是一个 IP 包，IP 包的有效载荷是一个 UDP 包，而 UDP 包的有效载荷是实际传输的数据。图 12.2 列出了这样一个帧的组成部分。

[3] 实际上，从互联网系统协会（Internet Systems Consortium）的网站 www.isc.org 可以得到一件显示这种 9 层扩展模型的体恤衫。

www.TopSage.com

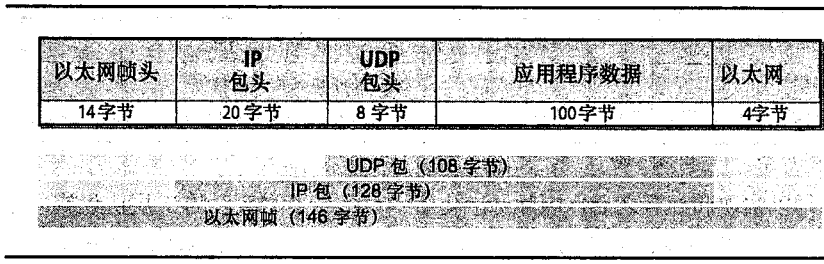


图 12.2 典型的网络包

在这里，“字节”一词指的是一个 8 比特的数据单位。以前，“byte（字节）”是一个更通用的术语，所以通常在 RFC 里看到的词是“octet（8 位组）”。

12.3.1 链路层

在这一节里，我们讨论在最低层的连网软件和网络硬件之间牵线搭桥的几个话题。

一、以太网组帧标准

链路层的主要任务是在数据帧上加上帧头，然后在帧之间插入分隔符。帧头包括帧的链路层寻址信息和校验和，分隔符可以确保接收方区分出上一个帧在哪儿终止，下一个帧从哪儿开始。加入这些额外比特位的过程就叫做组帧。

经常使用的有两种不同的 10Mbit/s 以太网组帧标准：DIX Ethernet II 和 IEEE 802.2 LLC SNAP[4]。以前，基于 Novell 的网络常用另一种称为 IEEE 802.2 LLC/SNAP 的标准。有些系统管理员可能会在他们的网络上发现还残留有这种标准。

一台主机使用的组帧方法是由其接口卡和接口卡的驱动程序共同确定的。在运行 Windows 的 PC 机上，可以选择您想使用的组帧类型，但在 Linux 上一般不能。从 Linux 的角度来看，这两种类型的组帧方式互操作性良好。另一方面，同一个网络上使用不同组帧方式的 Windows 主机却不能相互对话。作为系统管理员，除非您正在做的是在一个混合网络上进行低层的调试工作，否则一般情况下不必担心组帧不匹配。

二、以太网电缆连线和信号标准

各种速率的以太网（10Mbit/s、100Mbit/s、1Gbit/s 和现在的 10Gbit/s）选择什么样的连线方式通常都在 IEEE 制定的标准中做了规定。随着一种新技术的出现，就会有一种限于短距离的电缆得到批准。以后还会加上更廉价的介质、更宽松的限制。

参考第 14 章可以详细了解各种不同的以太网标准。另一个介绍以太网来龙去脉的参考资源是由 Charles Spurgeon 维护的网站 www.ethermanage.com/ethernet/。

三、无线连网

IEEE 802.11 标准尝试过定义无线链路的组帧和信号传输标准。遗憾的是，该标准最初相当模糊，而且包括几种没有充分确定的参数和选项。这里应该注意的一个互操作性问题是“转换”和“封装”的差异。

转换是将一个包从一种格式转换为另一种格式；封装是用所要求的格式包装包。Windows 系统倾向于默认进行封装，而 Linux 系统则进行转换，无线基站必须明确进行配置。如果您正在部署无线网络，那么必须保证基站以及与它们会话的工作站都按相同的模式运行。

[4] 链路层实际上又分为两个部分：MAC（Media Access Control，介质访问控制）子层和 LLC（Link Layer Control，链路控制）子层。MAC 层管理介质，把包送到导线上。LLC 层处理组帧。

www.TopSage.com

四、最大传输单位（*MTU*）

硬件规范和协议约定都可以对网络上传输的包的大小加以限制。例如，一个标准以太帧的有效载荷不能超过 1 500 字节。这一大小限制与链路层协议相关，叫做最大传输单位（Maximum Transfer Unit）或 MTU。表 12.2 列出了 MTU 的典型值。

表 12.2 各种网络链路层的 MTU

网 络 类 型	最大传输单位
以太网	1500 字节（采用 802.2 组帧时为 1492 字节）
FDDI	4470 字节（IP/FDDI 是 4352 字节）
令牌环	可配[a]
PPP 调制解调器链路	可配，常常是 512 或者 576 字节
PC 栈	可配，常常默认取 512 字节
点对点 WAN 链路（T1、T3）	可配，经常是 1500 或者 4500 字节

a. 常用的值是 552、1064、2088、4508 和 8232 字节。有时为了配合以太网而取 1500 字节。

在 TCP/IP 协议族里，为了符合某种特定网络链路的 MTU，IP 层会将包分片。如果包被路由通过几个网络，其中一个中间网络的 MTU 可能比源网络的 MTU 小。在这种情况下，将包转发到 MTU 较小的网络的路由器会进一步细分包，这个过程叫做分片。对于繁忙的路由器而言，分片是一项不受欢迎的工作。

TCP 协议可以确定通往目的地的路径上的最小 MTU，并且从始发端就使用这个 MTU 值。UDP 没有这么谦让，它乐于将额外的工作推给 IP 层。在 IPv6 协议中，新协议将分片变成了标准处理，中间路由器可以不再执行分片，但这要求检测出 MTU。

分片问题可能比较隐蔽。尽管能检测出路径的 MTU 应该就自动解决了 MTU 冲突问题，但是系统管理员偶尔也必须要介入一下。如果为 VPN（虚拟专用网络）采用隧道结构，那么应该看看穿过隧道的包的大小。开始它们一般为 1500 字节，但一旦加上隧道头部，就会变为 1540 字节左右，这样就必须进行分片处理了。将链路的 MTU 设置为一个较小数值可以避免分片，提高隧道网络的总体性能。请参考 **ifconfig** 手册页了解设置一个接口的 MTU 的方法。

12.3.2 包的寻址方式

与信件或者电子邮件相似，必须正确地对网络包寻址，以便能把它们送达它们的目的地。下面几种寻址方案要配合起来使用：

- 硬件用 MAC（medium access control，介质访问控制）寻址；
- 软件用 IP 寻址；
- 人用主机名寻址。

一台主机的网络接口通常有一个链路层 MAC 地址（将它与物理网络上的其他计算机区分开）、一个 IP 地址（在整个 Internet 上标识出该主机），以及一个主机名（供普通人使用）。

最底层的寻址机制是由网络硬件决定的。例如，以太网设备在制造的时候就分得了一个惟一的 6 字节硬件地址[5]。令牌环接口也有一个类似的地址，它也有 6 字节长。一些点对点的网络（如 12.13 节描述的 PPP 协议）根本不需要硬件地址，目的地的身份在建立链路的时候就指定好了。

[5] 至少从理论上说是惟一的。3Com 曾经在有不同类型的网络连接器的卡上重复使用了相同的以太网地址，它们想当然地认为用户只会定购一种网卡。这种投机取巧的做法在作介质转换的网点上造成了严重破坏，甚至在 3Com 的内部网络上也造成了破坏。在同一个网络上，MAC 地址冲突是致命的错误，而在由路由器隔开的网络上却不会带来问题。

www.TopSage.com

6 字节的以太网地址分为两部分：前 3 字节标识硬件的制造商，后 3 字节是厂家分配的惟一序列号。通常，系统管理员至少可以通过厂商 ID 表里的 3 字节标识符确定给网络制造麻烦的连网计算机的品牌。当前的厂商 ID 表可以从下面的地址查到：http://www.iana.org/assignments/ethernet-numbers。

这条信息过去都是定期在 RFC 系列里发布的，但现在不再按照这种方式发布了。RFC1700（1994）是最后一个 *Assigned Numbers*（分配编号）RFC。

以太网的硬件地址应该是永久分配、不能改变的。遗憾的是，有些网络界面卡能让您指定硬件地址。在这方面，无线网卡格外糟糕。请不要分配多播地址范围里的数值（第二位为奇数）或者使用其他特殊值。

Linux 能让您改变以太网接口的硬件地址，但是请不要这样做，它能破坏防火墙和某些 DHCP 实现。不过，如果您不得不替换一台坏机器或者网卡，而出于某些原因又必须使用原来的 MAC 地址（例如，您所有的交换机都要过滤它，或者您的 DHCP 服务器根据 MAC 地址来分配地址），这一特性就显出它的方便之处了。

硬件层的再上一层采用了 Internet 寻址（更通常的叫法是 IP 寻址）方式。每个网络接口都分得一个 4 字节 IP 地址[6]。IP 地址在全网内惟一[7]，且与硬件无关。在下一节我们将讨论 IP 寻址。

IP 地址和硬件地址之间的映射是在 TCP/IP 模型的链路层中实现的。在支持广播的网络上（即允许将包发送至“这个物理网络的所有主机”的网络），ARP 协议可以不需要系统管理员的帮助而自动发现双方的映射关系。参考 12.6 节了解有关 ARP 的更多知识。

由于 IP 地址很长，显得像个随机数字，所以人们往往很难记住它。Linux 系统可以把一个或者多个主机名与 IP 地址联系起来，这样用户可以键入 yahoo.com 来代替键入 216.115.108.245。这种映射关系可以用多种不同的方法建立，从一个静态文件（**/etc/hosts**）到 NIS 数据库系统再到世界范围内的域名系统 DNS。请记住，主机名只是 IP 地址为方便起见而有的一种缩写。

12.3.3 端口

IP 地址标识了计算机，或者更精确地说，标识了计算机上的网络接口。它们并不具有足够的特异性来确定特定进程或服务的地址。TCP 和 UDP 用“端口”的概念扩展了 IP 地址。端口是 16 比特位的数字，它对 IP 地址进行了补充，用它指定一个特定的通信通道。标准的服务，如电子邮件、FTP 和远程登录服务器都与**/etc/services** 文件里定义的“知名”端口相关。为了避免假冒这些服务，Linux 系统限制服务程序绑定到端口号低于 1024 的端口上，除非它们是以 root 身份运行的。（但是任何人都能和运行在低端口号上的服务进行通信，这项限制仅对端口控制起作用。）

12.3.4 地址类型

在 IP 层和链路层上都定义了几种不同类型的地址：

- 单播——代表一台主机（实际上是网络接口）的地址；
- 多播——标识一组主机的地址；
- 广播——包括本地网络上所有主机的地址。

多播寻址有助于实现视频会议系统这样的应用，在这些应用中，同一组包必须发送给所有的参加者。IGMP（Internet Group Management Protocol，网际组管理协议）用于建立和管理主机组，把它们

[6] 网络接口实际上可以有不止一个 IP 地址，但这只是在很少的特定环境下才用的专门配置。参考 21.3 节了解有关虚拟（Virtual）接口的更多信息。

[7] 这里撒了一个小谎，但它在大多数情况下都是对的。参考 12.4 节里关于 NAT 的讨论，了解非惟一 IP 地址的浅析。

www.TopSage.com

按一个多播目的地来对待。多播 IP 地址从 224～239 之间的一个字节开始。

广播地址使用一种通配形式的特殊地址（主机部分的二进制位都是 1）联络到本地网络上所有主机。

在现如今的 Internet 上，多播实际上还没有被采用，但正在积极开展几个研究项目。

12.4 IP 地址详解

IP 地址也叫做 Internet 地址，它有 4 字节长，分为网络部分和主机部分。网络部分标识地址所指的逻辑网络，主机部分标识在网络上的一台计算机。

按照惯例，IP 地址被写成十进制的数字，每个字节一个数字，各个数字之间用点号隔开。例如，我们的“boulder”这台计算机的 IP 地址是“128.138.240.1”。最左边的字节最重要，它始终是网络部分的成分。

若一个地址的第一字节是 127，则表示“环回（loopback）网络”，环回网络是没有实际硬件接口、只有一台主机的虚拟网络。环回地址 127.0.0.1 始终都指向当前主机。它的符号名称是“localhost”。

一个接口的 IP 地址和其他参数都可以用 **ifconfig** 命令设置。在 12.7 节有关于 **ifconfig** 的详细描述。

12.4.1 IP 地址分类

在历史上，IP 地址根据最左边那个字节第一个比特位的不同而进行“分类”。这种分类决定了地址的哪些字节是网络部分，哪些字节是主机部分。现在，路由系统用一个明确的掩码来指定网络部分，并且可以在地址的任意两个比特位之间进行划分，并不只局限于以字节为界。但是，在没有给出明确划分的时候，仍以传统的分类作为默认值。

A 类、B 类和 C 类表示常规的 IP 地址。D 类和 E 类地址用于多播和研究目的。表 12.3 介绍了各类地址的特征。一个地址的网络部分用 N 表示，主机部分用 H 表示。

表 12.3　　历史上的 Internet 地址分类

地址类型	第一个字节[a]	格　式	说　明
A	1～126	N.H.H.H	非常早期的网络，或者为 DoD（美国国防部）保留
B	128～191	N.N.H.H	大型网点，通常要划分子网，以前很难得到
C	192～223	N.N.N.H	容易得到，常常成组获得
D	224～239	—	多播地址，不是永久分配的
E	240～255	—	试验地址

a. 0 是特殊值，正规的 IP 地址第一个字节不用 0。127 为环回地址保留。

12.4.2 子网和网络掩码

对于单个物理网络而言，连接 100 台以上计算机的情况是少见的。所以 A 类和 B 类地址（分别允许一个网络上有 16 777 214 和 65 534 台主机）实际上是相当愚蠢和浪费的。例如，126 个 A 类网络占掉了多达一半的可用地址空间。

有这类地址的大多数站点都使用了一种名为子网划分的细分寻址方案，在这个方案里，可以借用地址的主机部分来扩展网络部分。例如，B 类地址的 4 字节通常解释为 N.N.H.H。如果子网划分把第

www.TopSage.com

3 字节分配为网络号，而不是主机号，那么地址就会解释为 N.N.N.H。划分子网的这种用法可以把单独一个 B 类网络地址转变为 256 个不同的类似 C 类的网络，每个网络都能支持 254 台主机。

可以用 **ifconfig** 命令把一个明确的“子网掩码”与一个网络接口联系起来，从而影响地址的重新分配。对应一个 IP 地址网络部分的各个网络掩码位被设置为 1，而主机位则设置为 0。例如，对 N.N.N.H 配置的网络掩码是十进制的 255.255.255.0，或十六进制的 0xFFFFFF00。**ifconfig** 通常用一个地址本来的分类来计算出哪些位是属于网络部分的。在明确设置了掩码时，就可以取代这种行为。参考 12.7 节了解有关 **ifconfig** 命令的更多信息。

网络部分和主机部分之间的分界不必落在字节的边界处。但是网络位必须相邻，并且必须在地址的高端[8]。

不在字节边界结束的网络掩码解读起来麻烦一些，通常把它们写成/XX 的形式，XX 是地址的网络部分的比特位数。这在有时叫做 CIDR（Classless Inter-Domain Routing，无类域间路由）记法。例如，网络地址 128.138.243.0/26 是指前 3 字节为 128.138.243 的 4 个网络中的第 1 个网络。其他 3 个网络的第 4 字节分别是 64、128 和 192。与这些网络关联的网络掩码是 255.255.255.192 或者 0xFFFFFFC0，二进制为 26 个 1 后接 6 个 0。图 12.3 更详细地说明了这些数字间的关系。

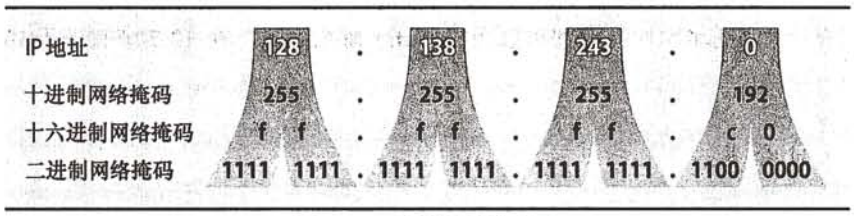


图 12.3 子网掩码基变换

一个/26 网络有 6 个比特（32–26=6）留下来给主机编号。因为 2^6 等于 64，所以网络具有 64 个可用的主机地址。但实际上它只能容纳 62 台主机，因为全 0 和全 1 的主机地址是保留的（它们分别是网络地址和广播地址）。

虽然在脑子里计算这些比特很容易搞混，但通过一些技巧可以简化它。每个网络的主机数目和网络掩码的最后一个字节的数值之和始终为 256：

最后一个网络掩码字节＝256－网络大小

例如，256–64＝192，它是前一个例子里网络掩码的最后一个字节。另一个算法是，某个实际网络地址（与网络掩码相反）的最后字节必须可以被每个网络上的主机数目整除。在当前的这个例子里就是这样的，最后一个字节分别是 0、64、128 和 192，它们都可以被 64 整除。

在这个例子里，子网网络地址中另外两位可以用数值 00、01、10 和 11。这样一来，网络 128.138.243.0/24 就分成了 4 个/26 网络：

- 128.138.243.0/26 （十进制 0 在二进制中是 **00**000000）；
- 128.138.243.64/26 （十进制 64 在二进制中是 **01**000000）；
- 128.138.243.128/26 （十进制 128 在二进制中是 **10**000000）；
- 128.138.243.192/26 （十进制 192 在二进制中是 **11**000000）。

每个地址的最后一字节中用粗体字显示的位是该字节中属于网络部分的位。

对于一个 IP 地址（如，128.138.243.100），在没有相关掩码的情况下我们不能给出网络地址和广播地址。表 12.4 显示了/16（默认为 B 类地址），/24（一个合理的数值），和/26（地址空间很紧缺时，一个较为实际的数值）的可能情况。

[8] 以前曾经允许 N.N.H.N 这样的配置，但并不普遍，现在已经不再允许了。

www.TopSage.com

表 12.4 IP 地址解读

IP 地址	子网掩码	网络地址	广播地址
128.138.243.100/16	255.255.0.0	128.138.0.0	128.138.255.255
128.138.243.100/24	255.255.255.0	128.138.243.0	128.138.243.255
128.138.243.100/26	255.255.255.192	128.138.243.64	128.138.243.127

Krischan Jodies 编写了一个方便的脚本，这个脚本叫做 IP 计算器（IP Calculator），它可以帮助进行二进制/十六进制/掩码的运算。从 www.jodies.de/ipcalc 可以下载到它。**ipcalc** 显示了您可能应该了解的所有关于一个网络地址及其网络掩码、广播地址、主机方面的知识。网上的版本是个 tar 的压缩文件，而 Debian 和 Ubuntu 默认就带一个 **ipcalc** 的命令行版本。

下面是一些 IP 计算器输出的例子，可以通过格式提供一点帮助：

```
Address:   24.8.175.69          00011000.00001000.10101111.01000101
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111.00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000.11111111
=>
Network:   24.8.175.0/24        00011000.00001000.10101111.00000000 (Class A)
Broadcast: 24.8.175.255         00011000.00001000.10101111.11111111
HostMin:   24.8.175.1           00011000.00001000.10101111.00000001
HostMax:   24.8.175.254         00011000.00001000.10101111.11111110
```

这些输出既给出了易于理解的地址版本，又给出了“剪切和粘贴”的版本。非常酷。



Red Hat 里有一个程序，也叫做 **ipcalc**，它的功能很差，大多数计算都要假定 IP 地址是 A 类、B 类或者 C 类。

最初定义 IP 子网划分的 RFC（RFC950）不允许使用第一个或者最后一个子网（全为 0 或者全为 1）。在/26 网络的例子中，这个规则排除了一半子网：0 子网和 192 子网。除了 Novell 和 Cisco 之外，所有地方都忽略这个 RFC（在 Cisco 早期的 IOS 操作系统版本中，必须用 `ip subnet zero` 命令明确启用子网 0。在 12.0 及其以后的版本中，子网 0 是默认可用的）。

虽然这个 RFC 的意图很好，但它却是错误的。不允许子网 0 出现是因为人们认为如果一个子网的地址与没有划分子网的网络地址相同，不能区分开，就有可能产生混淆。但是事实证明这种担心是没有根据的，现在全 0/全 1 子网都得到了广泛应用。不能为全 0 或全 1 的应该是主机部分。

网络地址和广播地址从每个网络中占用了两个主机，所以最小的有意义的网络应该有 4 台主机：两个实际主机——通常在一条点对点链路的两端——以及网络地址和广播地址。为了使主机有 4 个值，需要在主机部分有 2 位，所以这样的网络应该是网络掩码为 255.255.255.252 或者 0xFFFFFFFC 的一个/30 网络。不过，/31 网络被当作一种特例（参考 RFC3021），它没有网络或者广播地址，两个地址都供主机使用，它的网络掩码是 255.255.255.254。

虽然网络上的主机可能会同意使用作了子网划分的地址，但世界上的其他地方并不知道这一点，于是仍然按照原本的 IP 地址类处理地址[9]。在我们给出的 128.138.243.100 这个例子中，并没有向外面的世界通告每个子网，而只需通告到单个 B 类网络就可以了。一旦一个包到达了划分了子网的区域，它的目的地址就会用本地网络掩码重新解释，从而发现实际的目标网络，并把这个包路由至它的确切目的地。

12.4.3 IP 地址危机

大约在 1992 年，Internet 界就认识到，最初的地址分配方案存在 3 个根本性的问题：

[9] 另一个以简单为名，但却不全面的假话，请看 12.4.4 节关于 CIDR 的讨论。

www.TopSage.com

- 首先，到 1995 年年中，我们就会用完 B 类地址——一般的大单位最希望得到这类地址；
- 其次，Internet 主干结点的路由表增长得非常大，以至于在路由器的内存中装不下这些表；
- 最后，IP 地址是按“先来先得”的方式分配的，不具有地域性；也就是说，数字上相邻的两个地址可能在同一个机构里，也可能在不同的大洲，试想一下，如果邮政编码是按照这种随机的方式分配的，将会造成多大的混乱。

为了解决这个问题，人们先后推出了两种解决办法：一个是近期的解决方法，另一个是长期的解决方法。

近期的解决方法是 CIDR（Classless Inter-Domain Routing，无类域间路由），它是另一种处理现有的 4 字节地址空间的方法，它通过考虑数字之间的衔接关系，更有效地利用了可用的地址，并且简化了路由表。我们将在下一节详细讨论 CIDR。

长期的解决方法是 IPv6，它是对 IP 协议的修订。IPv6 将地址空间扩展至 16 字节，并且融进了最近 25 年使用 IP 协议的其他经验教训。它删除了 IP 协议中一些由经验证明没有多少价值的功能，使协议有可能更快、更容易地实现。IPv6 还将安全和身份验证功能集成到基本协议中，消除了在中间路由器上做分片的可能性。

IPv6 仍处于开始部署的早期阶段，但 CIDR 已经全面投入应用好多年了。CIDR 得到 Internet 主干和路由设备主要制造商的支持和使用。NAT（Network Address Translation，网络地址转换）是重用 IP 地址的方案，它在降低 IP 地址的需求量方面也起到了巨大的作用，我们将在 12.4.6 节介绍这个方案。

IPv6 的复杂性、CIDR 和 NAT 的有效性以及 Internet 已经运行良好的惯性都表明，还需要很长一段时间才能够转移到 IPv6 上。虽然许多应用软件和操作系统自身都已经支持 IPv6 了，但是在许多美国人的观念里，单是转移到 IPv6 的成本这一项就已经很破费了。尽管如此，在 Linux 发行商和志愿开发人员这一边仍投入了更多的努力，以保证在网络切换到 IPv6 的时候，Linux 准备就绪。

在国际舞台上，IPv4 地址更为紧缺，所以 IPv6 在那里更受欢迎。而在美国，可能某个需要 IPv6 的新应用程序来推进，例如，一种新一代的手机把一个 IPv6 地址和一个电话号码对应起来。VoIP 系统也会从电话号码和 IPv6 地址之间更为密切的对应关系中受益。

12.4.7 节有关于 IPv6 寻址的更多内容。

12.4.4 CIDR：无类域间路由

CIDR 是在 RFC1519（1993 年 9 月）里定义的，它废除了以前由一个 IP 地址的网络部分所确定的分类系统。CIDR 与子网划分相似，是子网划分方法的直接扩展。CIDR 要依赖于一个明确的网络掩码来确定地址的网络部分和主机部分之间的边界。但与划分子网不同的是，出于路由选择的目的，网络部分比地址暗示的分类所暗指的网络部分小。使用较短的网络掩码可以聚集若干个网络。因此，CIDR 有时叫做超网（supernetting）。

使用 CIDR 就可以将若干 C 类网络分配给一个网点，而不要求 Internet 对各个网络具有单独的路由表项。还可以为这个网点分配某个 A 类或 B 类地址的一个子空间。例如，假设某个网点已获得 8 个 C 类地址，从 192.144.0.0 到 192.144.7.0（CIDR 的记法为 192.144.0.0/21）。这个网点在内部可以将它们用作：

- 一个长度为/21 的网络，2 046 台主机[10]，网络掩码为 255.255.248.0；
- 8 个长度为/24 的网络，各自具有 254 台主机，网络掩码为 255.255.255.0；
- 16 个长度为/25 的网络，各自具有 126 台主机，网络掩码为 255.255.255.128；
- 32 个长度为/26 的网络，各自具有 62 台主机，网络掩码为 255.255.255.192 等。

[10] 最初在 RG-11 同轴电缆上的以太网允许一个网络上最多有 1 024 台主机。采用现在的交换机，还有可能构造实际上更大的网络（但不是非常可取）。

www.TopSage.com

混合和匹配不同的子网长度也是可能的，只要所有部分可以不重叠地拼在一起即可。这叫做变长子网划分：例如，一个分配了地址 192.144.0.0/21 的 ISP 可以为 PPP 拨号上网的客户定义一些/30 网络，为大客户定义一些/24 网络，为较小的用户定义一些/27 网络。

在像这样混合和匹配子网的时候，某个特定网络上的所有主机必须配置相同的网络掩码。您不能告诉网络上的一台主机它是/24 网络，而告诉同一个网络上的另一台主机它是/25 网络。

CIDR 的优点和价值要从 Internet 的角度来看，没有必要为这些地址保留 256、128 甚至 32 条路由表项。它们都指向同一机构，所有的包都需要送到相同的地方。对于地址 192.144.0.0/21 而言，一条路由项就足够了。此外，CIDR 让分配 A 类和 B 类地址的一部分或者几部分的工作变得容易了，因而成倍地增加了能够使用的地址。

随着 CIDR 的出现，系统管理员已经能够熟练地进行二进制和十六进制运算了，他们发现 Linux 程序 **bc** 可以用 **ibase** 和 **obase** 命令，在任一进制下做数学计算[11]。您可以把表 12.5 用作一张速查表，省去计算的麻烦。

表 12.5　　各种不同长度的网络掩码的网络配置

长　度[a]	主机比特位	主机/网[b]	十进制网络掩码	十六进制网络掩码
/20	12	4094	255.255.240.0	0xFFFFF000
/21	11	2046	255.255.248.0	0xFFFFF800
/22	10	1022	255.255.252.0	0xFFFFFC00
/23	9	510	255.255.254.0	0xFFFFFE00
/24	8	254	255.255.255.0	0xFFFFFF00
/25	7	126	255.255.255.128	0xFFFFFF80
/26	6	62	255.255.255.192	0xFFFFFFC0
/27	5	30	255.255.255.224	0xFFFFFFE0
/28	4	14	255.255.255.240	0xFFFFFFF0
/29	3	6	255.255.255.248	0xFFFFFFF8
/30	2	2	255.255.255.252	0xFFFFFFFC

a. 网络长度+主机的比特位数始终等于 32，这是因为我们在划分固定长度的 32 位 IP 地址。

b. 数学好的人会发现，每个网络上的主机数目是 2^主机比特位数–2；–2 反映出全 0 和全 1 的主机地址是特殊地址。

在 1993 年引入 CIDR 的时候，主干网的路由表大约包括 20 000 条路由。自从.com 网络公司爆炸性发展之后，Internet 缓慢而稳定地将路由表的大小增加到了 2006 年的 250 000 条路由[12]。

地址空间中仍有一个没有聚类的区域，叫做 192 泥潭（以及一些在 199 和 205 范围内的较小泥潭），它由早期的 C 类地址组成，这些 C 类地址的所有者不能将它们聚类也不希望归还这些地址然后重新编号。在这一方面美国尤其糟糕。欧洲和亚洲启动得稍晚，所以从美国的错误中吸取了经验和教训，在分配地址方面的工作做得比美国好得多。有未聚类的 192 网络的地方应该将其还给 ARIN（American Registry for Internet Numbers，美国 Internet 编号注册机构），然后从它们的 ISP 那里获取一个新的地址块。遗憾的是，重新编号的代价（至少在 IPv4 空间里）太大，使得大多数地方都没有这样做。

虽然 CIDR 的目的只是作为一个过渡的解决方案，但事实证明，它有足够的能力，可以解决在可预见的一段时间里碰到的 Internet 增长问题。事实上，CIDR 和 NAT 的组合取得的效果非常好，以至于我们开始怀疑是否真的需要一个新的 IP 协议。现在已经为 IPv6 规范和 IPv6 的原型实现进行了大量

[11] 请注意，不要把你自己逼向一个死角，这个问题留作读者的练习题。

[12] 请看 bgp.potaroo.net，了解 BGP 路由表的更多知识。

www.TopSage.com

的工程工作。如果浪费掉这些工作将令人难堪，但 IPv6 的大规模部署可能需要有个必备的 IPv6 新应用，这种应用只能采用 IPv6 寻址机制，要么就是需要微软作出废除 IPv4 的决定。

12.4.5 地址分配

在 Internet 的早期，各个独立的站点向 InterNIC（Internet Network Information Center ，Internet 网络信息中心）申请地址空间。如今在美国，ARIN 和 LACNIC 已经代替了 InterNIC。现在只有每年分配出大量地址空间的 ISP 才有资格向 ARIN 申请 IP 地址空间。其他所有地方必须向它们的 ISP 申请。

只有网络编号是正式分配的，站点必须定义它们自己的主机编号以构成完整的 IP 地址。您可以按照自己喜欢的任意方式再对分配给您的地址空间进行细分。

在管理上，ICANN（the Internet Corporation for Assigned Names and Numbers，Internet 名称和编号分配组织）将地址块委托给 5 个区域性 Internet 注册机构，这些地区性权威机构负责在本地区内把子地址块分配给 ISP（请看表 12.6）。接着这些 ISP 再分割它们的地址块，分配给各个客户。只有大型 ISP 才会直接和 ICANN 授权的区域性地址注册机构之一打交道。

表 12.6 区域性 IP 地址注册机构

名 称	Web 地址	覆 盖 区 域
ARIN	www.arin.net	北美洲，加勒比海部分地区
APNIC	www.apnic.net	亚洲/太平洋地区
AfriNIC	www.afrinic.net	非洲
LACNIC	www.lacnic.net	中南美洲，加勒比海部分地区
RIPE	www.ripe.net	欧洲及周边地区

从 ICANN 到 5 个区域性 NIC，再到国家或地区 ISP 的委托管理已经考虑到主干路由表中的聚类问题。ISP 的用户已经分配了这个 ISP 拥有的地址块内的地址空间，就不需要主干上的单独路由项。一条指向 ISP 聚类块的路由表项就足够了。

最开始，地址空间的分配不是非常公平。美国政府为自己保留了大约一半的地址空间，只把相当小的一部分地址块分给欧洲和亚洲。但欧洲和亚洲的地址空间管理工作做得比美国好。www.caida.org/analysis/learn/ipv4space 上的地址空间图谱相当清楚地揭示了这个事实，它显示出 IP 地址空间总体、已经分配的部分、有路由的部分（因此是可达的部分）以及在美国的两个主要交换点观测到流量的地址。

12.4.6 私用地址和 NAT

另一个暂时解决地址空间短缺的方法是使用 RFC1918（1996 年 2 月）中描述的私用 IP 地址空间。在 CIDR 时代，站点通常从 Internet 服务提供商（ISP）得到 IP 地址。如果一个站点想更换其 ISP，那么可能因为考虑到重新为其网络分配编号所带来的成本而退缩。只要是用户，ISP 就会给它地址空间。如果站点想选择一个不同的 ISP，那么它必须说服旧的 ISP 为它保留地址，还要说服新的 ISP 把旧地址正确地路由到新位置。一般情况下，ISP 并不想惹上这些麻烦问题，所以会要求用户重新编号。

使用 ISP 分配的地址的另一个方法是，使用决不会让您的 ISP 知道的私用地址。RFC1918 留出了一个 A 类网络、16 个 B 类网络、以及 256 个 C 类网络，这些网络从不在全网内分配，但可以在任意站点内部使用。这里的关键之处在于，带有那些地址的包必须永远不能溜出来跑到 Internet 上。您应

www.TopSage.com

大家网 TopSage.com

该在边界路由器上进行过滤，保证做到这一点。如果有一些包溜了出去，您应该找出放它们出去的错误配置。

表 12.7 显示了为私用寻址保留的网络号（“CIDR 范围”这一列用更加简洁的 CIDR 记法说明了每类地址的范围，它并没有任何其他意思）。

表 12.7 为私用而保留的 IP 地址

IP 分类	从	到	CIDR 范围
A 类	10.0.0.0	10.255.255.255	10.0.0.0/8
B 类	172.16.0.0	172.31.255.255	172.16.0.0/12
C 类	192.168.0.0	192.168.255.255	192.168.0.0/16

站点可以从这组网络中选取最适合它们各自机构的网络的大小。

为了让使用这些私用地址的主机与 Internet 对话，站点的边界路由器要运行一个名为 NAT（Network Address Translation，网络地址转换）的系统。NAT 截取带有只在内部使用的地址的包，然后用一个真实的外部 IP 地址，也许还有一个不同的源端口号，来改写它们的源地址。它还维护着一张内外地址/源端口对之间的映射表，这样当应答包到达的时候就可以进行反向转换了。

NAT 使用了端口号映射，这样就能让若干个对话复用同一个 IP 地址，从而让许多内部主机可以共享一个外部地址。在有些情况下，一个站点只能分配一个“真实”的 IP 地址。

使用 NAT 的站点仍然要从它的 ISP 申请地址空间，但由此获得的大多数地址都用于 NAT 映射，并不分配给单个主机。如果以后站点想选择另一家 ISP，只要改变边界路由器和它的 NAT 配置即可，并不需要改变单个主机的配置。

NAT 在消费行业和企业界都得到广泛应用。即便许多站点都更倾向于将 NAT 任务交给他们的路由器或者网络连接设备去完成，但是也可以让一台 Linux 主机执行 NAT 功能。参考本章针对各发行版本的节了解详细情况。出于某些原因，Linux 世界过去把 NAT 叫做“IP 伪装（IP masquerading）”。不过，随着 2.4 内核的出现，Linux 也开始把它叫做 NAT 了。

不正确的 NAT 配置会让私用地址空间的包漏到 Internet 上。虽然这些包会到达它们的目的地，但是应答包却不能返回来。CAIDA[13]是一个对主干网络进行各项测量的组织，它发现主干上 0.1%～0.2% 的包不是带有私用地址就是带有错误的校验和。虽然这看起来只是一个很小的比例，实际上也是如此，但是这意味着在 MAE-West（一个主要的公共交换站，若干不同的 ISP 在这里汇集交换流量）的一条繁忙的电路上，每 10 分钟就有 20 000 个这样的包。在 www.caida.org 上还有其他一些有趣的统计和网络测量工具。

NAT 的缺点之一（或者可能也是它的一个优点）是，Internet 上的任何主机都不能直接与站点的内部计算机连接。有些实现（例如，Linux 和 Cisco PIX）会让您配置支持直连某些特定主机的“隧道”。

另一个问题是一些应用在包的数据部分嵌入了 IP 地址，这些应用会被 NAT 阻止或者混淆。这样的例子包括某些路由协议、RealVideo 和 SHOUTcast 这样的流媒体程序、PORT 和 PASV 这样的 FTP 命令、ICQ 即时消息传送以及许多游戏。NAT 有时也会打断 VPN（virtual private networks，虚拟私用网络）。

使用 NAT 和 RFC1918 地址的大型企业必须建立某种形式的中央协调机制，让属于各自独立的部门或者管理小组的所有主机都有惟一的 IP 地址。当一家采用 RFC1918 地址空间的公司吞并或者合并另一家采用同样做法的公司时，这种情形就可能变得复杂起来。合并后的单位里的一些部门往往必须重新编号。

[13] CAIDA，念作“kay duh”，是 Internet 数据分析合作协会（Cooperative Association for Internet DataAnalysis），它位于 UCSD（加州大学圣地亚哥分校）校园内的圣地亚哥超级计算机中心（San Diego Supercomputer Center），网址为 www.caida.org。

www.TopSage.com

NAT 隐藏了内部结构。这个保密性似乎是安全方面的赢家，但安全人士声称 NAT 实际上对安全没有帮助，而且肯定不能代替对防火墙的需要。它还会阻止对 Internet 的大小和拓扑结构的测量。

至少已经有一种工具被开发出来，用于识别使用 NAT 的主机。这个工具的算法把已知的操作系统 TTL 值同网络上预期的 TTL 值关联起来。它们不匹配的时候，源 IP 地址就确定为一个 NAT 设备。这一思想从理论上说有效，但在实际中实现起来代价高，难处理。参考 www.sflow.org/detectNAT 了解这个方法的更多知识。

12.4.7 IPv6 寻址方式

IPv6 地址的长度为 128 比特。这些长地址的最初目的是为了解决 IP 地址枯竭的问题。但是现在利用它们来解决路由、可移动性和区域性等问题。

IP 地址从来没有像电话号码或者邮政编码那样按地理位置聚集。现在，根据对 IPv6 地址空间分段的提议，它们至少要按 ISP 聚集。IPv6 地址的网络部分和主机部分之间的界线固定在/64，网络的公共拓扑结构和站点本地的拓扑结构之间的界线固定在/48。表 12.8 列出了 IPv6 地址的各个部分。

表 12.8 IPv6 地址的各个部分

完整的 IPv6 地址（128 位）

ISP 前缀	子网	主机标识符
45位	16位	64位

地址类型 3位

位	首字母缩写	转　换
1～3	FP	格式前缀，地址类型，例如单播
4～16	TLA ID	最高层的聚类 ID，如主干 ISP
17～24	RES	保留，供以后使用
25～48	NLA ID	下一层聚类 ID，例如，区域性 ISP 和站点 ID
49～64	SLA ID	站点层的聚类 ID，如本地子网
65～128	INTERFACE ID	接口标识符（MAC 地址加填充）

在这些片段中，只有 SLA ID 和 INTERFACE ID 属于主机和它的站点。其他部分都由上游 ISP 提供。SLA 指定了一个本地子网。64 位接口 ID 标识主机的网络接口，在一般情况下，它包括 48 位的 MAC 地址，中间带有十六进制数字 0xFFFE。在 MAC 地址中有一个特殊的位（第一个字节，从左起，从 0 位开始的第 6 位），叫做通用/本地位，必须补足这一位（参考 RFC2373）。这个方案可以为主机自动编号，这对于系统管理员来说是一个很好的功能，因为只需要管理子网即可。

在 IPv6 里，MAC 地址在 IP 层上可见，这既有好处又有坏处。接口卡的品牌和型号在 MAC 地址的前半部分编码，所以从针对某个特定结构的代码上，黑客能得到很大的帮助。这一信息的可见性也令提倡保护隐私权的人们担心。IPv6 的开发人员指出，实际上并不要求站点采用 MAC 地址，他们可以随意用各种地址作为主机地址。现在已经提出了一种方案，在地址的本地部分里加入一个随机的标记。需要进行处理的比特真是太多了！

在另一方面，分配 IPv6 地址应该比分配 IPv4 地址简单，因为只需要知道子网的地址就可以了。主机可以对自身进行配置，至少在理论上说是可以的。

格式前缀标识了 IPv6 地址的类型：单播（unicast）、多播（multicast）或者任意传送（anycast）。单播地址把 FP 设置为 001（二进制）。TLA 和 NLA ID 标识最高层的 IP 主干运营商以及向上链到主干提供商的本地 ISP。

www.TopSage.com

现在大多数厂商都已经支持 IPv6 协议栈了。几乎所有的 Linux 发行版本本身都支持 IPv6，而网络硬件已经支持 IPv6 将近 10 年了。

ARIN 只向大型 ISP 分配 IPv6 空间，或者给计划在未来分配出大量 IPv6 地址块的 Internet 注册机构分配 IPv6 空间。这些机构接着可以向其下游客户分配更小的空间。费用最低为每年 1 250 美元，最高为每年 36 000 美元。ARIN 扩大了面向有效会员的免费范围。

下面是一些有用的 IPv6 信息资源：

- www.ipv6tf.net——IPv6 信息门户；
- www.ipv6.org——FAQ 和技术信息；
- www.ipv6forum.com——市场营销人士和 IPv6 倡导者的论坛。

IPv6 的一个主要优点在于它解决了重新编号的问题。在 IPv4 里，ISP 要将地址空间分配给用户，但这些地址是不能移动的，当用户离开某个 ISP 的时候，他们必须交回他们的地址，然后用他们从新 ISP 得到的地址重新编号。而在 IPv6 里，新 ISP 给您一个地址前缀，您只要预先考虑地址的本地部分就可以了，可能只在边界路由器上考虑这些就可以了。这个方案与 IPv4 寻址的 NAT 方案类似，但是它不存在 NAT 方案中的任何小问题。

为了促进从 IPv4 到 IPv6 的迁移，人们已经提出了各种方案，包括使用 NAT，在数据包通过现有 IPv4 基础设施里的隧道时隐藏 IPv6 地址。

12.5 路由选择

路由选择（routing，也叫做路由、选路）是引导一个包通过网络迷宫，从它的源地址到达目的地址的过程。在 TCP/IP 系统中，这类似于在一个陌生的国家里问路。您问的第一个人可能把您指引到正确的城市。当离目的地比较近的时候，下一个人就能告诉您怎样到达正确的街道。最后，距目的地很近了，这时就有人可以告诉您您想找的那幢大楼了。

TCP/IP 路由信息采用了规则（“路由[route]”）的形式，如“为了到达网络 A，要经过计算机 C 发送包”。同时还存在一条默认路由，它告诉您怎样处理要送往没有明确路由的网络的包。

路由信息保存在内核的一张表里。表中的每一项都有一些参数，其中包括每个列出的网络的网络掩码（这个参数以前是可选的，但是现在如果默认的网络掩码不正确，就必须给出）。为了把一个包路由至特定的地址，内核会选择与之匹配的路由中最确切的一条（即带有最长网络掩码的路由）。如果内核没有找到适当的路由，又没有默认路由的话，那么它就向发送方返回一个“网络不可达（network unreachable）”的 ICMP 错误消息。

“路由”一词通常表示两种截然不同的意思：

- 在路由表里寻找一个网络地址，以便将一个包转发到它的目的地；
- 首先建立路由表。

在这一节里，我们考察转发功能，并介绍怎样手工从路由表中删除路由或者向路由表里增加路由。我们把建立和维护路由表的路由协议这个更复杂的话题留给第 13 章介绍。

12.5.1 路由表

可以用 **netstat -r** 命令检查一台计算机上的路由表。用命令 **netstat-rn** 避免做 DNS 查询，并把所有的信息以数字地址形式输出。我们将从 19.4 节开始详细地讨论 **netstat** 命令，但是这里给出的简短例子可以让您对路由有一个比较形象的概念。这台主机有两个网络接口：在 132.236.227.0/24 网络上的 132.236.227.93（eth0），和在 132.236.212.0/26 网络上的 132.236.212.1（eth1）：

www.TopSage.com

```
$ netstat -rn
Kernel IP routing table
Destination     Genmask         Gateway         Fl   MSS   Iface
132.236.227.0   255.255.255.0   132.236.227.93  U    1500  eth0
default         0.0.0.0         132.236.227.1   UG   1500  eth0
132.236.212.0   255.255.255.192 132.236.212.1   U    1500  eth1
132.236.220.64  255.255.255.192 132.236.212.6   UG   1500  eth1
127.0.0.1       255.255.255.255 127.0.0.1       U    3584  lo
```

Destination（目的地）域通常就是一个网络地址，Gateway（网关）必须是一个主机地址。例如，第 4 条路由说明，为了到达网络 132.236.220.64/26，必须通过接口 eth1 把包发往网关 132.236.212.6。第 2 行是一条默认路由，没有明确指出发往这 3 个网络中任何一个（或该计算机本身）的包将发送到默认网关主机，132.236.227.1。主机可以将包只路由到直接连接到和他们相同的网络上的网关计算机。

路由表可以静态或动态地配置，也可以结合这两种方法进行配置。静态路由是用 **route** 命令明确地输入的路由。只要系统还在运行，静态路由就应该在路由表里，它们通常是通过一个系统启动脚本在系统引导时设置的。（参考 12.7.4 节了解有关 **route** 命令的更多信息。）例如，Linux 命令：

```
route add -net 132.236.220.64 netmask 255.255.255.192 gw 132.236.212.6 eth1
route add default gw 132.236.227.1 eth0
```

会加入前面 **netstat -rn** 显示的第 4 和第 2 条路由（显示的第 1 和第 3 条路由是在配置 eth0 和 eth1 接口的时候，由 **ifconfig** 命令加上的）。

最后一条路由也是在系统引导时加入的。它配置了一个名为 loopback（环回接口）的伪设备。环回接口不让主机发送给自己的包跑到网络上去，而是在内核中直接从网络输出队列传输到网路输入队列。

在一个稳定的本地网络里，静态路由是一种高效的解决方案。它易于管理，并且很可靠。但是，它要求系统管理员确切地知道系统引导时网络的拓扑结构，而且这个拓扑结构不会经常改变。

在局域网中的多数机器只有一条路径连接到网络的其余部分，所以路由问题十分简单。在系统引导时加入的一条默认路由，就足够指出出去的路径了。使用 DHCP（参考 12.9 节）得到自身 IP 地址的主机也能用 DHCP 得到一条默认路由。

对于更复杂的网络拓扑结构来说，需要动态路由。一般情况下，动态路由是由维护和修改路由表的守护进程执行的。不同主机上的路由守护进程间进行通信，从而发现网络的拓扑结构，并且计算出如何到达远程目的地。现在可以使用若干种路由守护进程。

12.5.2 ICMP 重定向

虽然在通常情况下，IP 协议与路由信息的管理并没有关系，但它确实定义了一个小小的故障控制功能，称为 ICMP 重定向（ICMP redirect）。当一台路由器把一个包转发至一台计算机，而这台计算机却又在最初接收这个包的网络上的时候，就出现了明显的错误。由于发送方、路由器和下一跳的路由器都在同一网络上，所以包应该以一跳而不是两跳转发。路由器可以得出结论：发送方的路由表是不准确或者不完整的。

在这种情况下，路由器可以用一个 ICMP 重定向包把发送方的问题通知给它。实际上，一个重定向包说："您不应该将发往主机 *xxx* 的包发送给我，您应该把它们发送给主机 *yyy*。" ICMP 协议允许将重定向包既发送给单个主机地址也发送给整个网络。但是，许多实现只产生主机重定向包，现在，网络重定向包还没有多大用处，因为它们只能应用到 A、B 或者 C 类网络上。

www.TopSage.com

根据接收到的重定向包，发送方就会更新它的路由表，这样一来，将来那些发往这个地址的包就能走更直接的路径。在出现多播的早期时代，有些系统会响应多播包而生成 ICMP 路由重定向包。现代系统已不存在这个问题。

标准的 ICMP 应用场合并不包括身份认证的步骤。您的路由器会收到一个声称来自其他某个声誉良好的路由器的重定向包，指导您将流量发送到其他地方。您是否应该采纳呢？请注意，重定向实际上引发了一个安全问题。一般情况下，Linux 内核（出于安全原因）和 Cisco 路由器（因为它们是路由器）都会忽略重定向包。让那些不可信的主机修改您的路由表可不是件好事。

在 Linux 下，**/proc** 目录树下的变量 **accept_redirects** 控制着 ICMP 重定向包的接收。参见 12.10 节了解检查和重设这个变量的指令。

12.6 ARP：地址解析协议

虽然在通常情况下，IP 包是根据 IP 地址发送的，但实际上必须用硬件地址在网络链路层上传送数据[14]。ARP，即地址解析协议，能发现与特定 IP 地址相关的硬件地址。它可以在所有支持广播的网络上使用，但最常见的还是按以太网的方式来介绍它。

如果主机 A 想把一个包发送到同一个以太网上的主机 B，那么它用 ARP 来发现 B 的硬件地址。当 B 与 A 不在同一个网络上的时候，主机 A 用 ARP 来发现目标地址为 B 的包应发往的下一跳路由器的硬件地址。因为 ARP 使用广播包，这个包不能跨越网络[15]，所以它只能用于寻找和发送主机所在的本地网络直接连接的计算机的硬件地址。

每个计算机都在它们的内存里维护一张叫做 ARP 缓存的表，这张表保存了近期 ARP 查询的结果。正常情况下，主机在启动后很快就能发现它所需要的许多地址，所以 ARP 不会造成大量的网络流量。

ARP 通过广播[16]这种形式的包“有人知道 128.138.116.4 的硬件地址吗？”来发挥作用。被搜索的计算机会识别出这是自己的 IP 地址，然后返回一个应答：“是的，这是分配给我的一个网络接口的 IP 地址，而相应的以太网地址是 8:0:20:0:fb:6a。”

初始查询包括请求方的 IP 地址和以太网地址，这样正被搜索的计算机可以在不发送它自己的 ARP 查询的情况下给予应答。这样，这两台计算机只要一次包交换就可以知道彼此的 ARP 映射了。其他监听到请求方的初始广播的计算机也可以记录下它的地址映射。这种被动的 ARP 流量检测有时叫做探听（snooping）。

arp 命令可以检查和处理内核的 ARP 缓存。**arp** 一般用于增加或者删除一条缓存项，它也可以刷新缓存表或者显示它。**arp -a** 会显示 ARP 缓存的内容。例如：

```
redhat$ /sbin/arp -a
sprint-gw (192.168.1.254) at 00:02:4B:5B:26:45 [ether] on eth0
inura-local.toadranch.com(192.168.1.101)at 00:04:76:37:AE:7E[ether]on eth0
```

一般情况下，**arp** 命令只有在调试和涉及特殊硬件的时候有用。一些设备还没有聪明到能使用 ARP 协议的程度（例如，网络打印机或者特殊用途的图形显示器）。为了支持这样的设备，您可能需要配置其他计算机，作为这些功能欠缺的硬件的代理 ARP 服务器（proxy ARP server）。通常也可用 **arp** 命令来实现（使用**-s** 标志）。例如：

[14] 点到点链路除外，在这种链路上，目的地址的标识有时是隐含的。

[15] 路由器常常可以配置成让大量广播包涌入其他网络，请不要这样做。

[16] ARP 采用底层链路层的广播约定，而不是 IP 广播的约定。

www.TopSage.com

```
# /sbin/arp -s printer.toadranch.com 00:50:04:ce:ef:38
```

Linux 的 2.4 及以后版本的内核不支持面向整个子网的 proxy ARP 服务，但在有一条路由，而接口又配置了转发包的功能时，它会自动充当 proxy ARP 服务器。

如果一个网络上的两台主机使用了相同的 IP 地址，其中一台的 ARP 表项是正确的，而另一台的 ARP 表项是错误的。您可以使用 **arp** 命令找到这台出错的计算机。

有时需要将硬件地址转换为 IP 地址。许多功能不足的硬件（如无盘工作站、网络计算机、打印机等）需要在启动时执行这个转换。计算机可以查询中央服务器找到它自己的地址，而不是在一个配置文件中直接写入一个 IP 地址。

至此我们讲完了有关连网技术的背景材料。在下面的几节里，我们讲述在为本地网络和 Internet 配置 Linux 主机时涉及的若干问题。

12.7 将一台主机加入网络

只需几步，就能把一台新主机加入到一个现有的局域网中，但有些厂商把您必须修改的文件给隐藏了起来，因而常常使这一工作变得困难起来。别的一些厂商则提供了一个设置脚本，提示您输入所需的连网参数，这个脚本不错，但当您需要取消所做的某些操作或者移走一台主机的时候，就不是很好用了。在将一台连接到 Internet 的新机器连接到网络之前，您应该确保它的安全（见第 20 章），以便不会把黑客引到本地网络上来。

把一台主机加入本地网络的基本步骤是：

- 分配惟一的 IP 地址和主机名；
- 设置新主机在启动时配置它的网络接口；
- 设置一条默认路由，或许还有更多的路由配置；
- 指向一台 DNS 名字服务器，让它能访问 Internet 的其余部分。

当然，您也可以在上述步骤中加上调试步骤。在进行了任何可能影响引导的改动之后，您都应该再次引导系统以证实这台计算机设置正确。六个月之后，当电源发生故障，然后计算机再启动不了的时候，会很难记起您所作过的改动，而可能就是它造成了这个问题（参考第 19 章）。

如果您的网络采用了 DHCP（Dynamic Host Configuration Protocol，动态主机配置协议），那么 DHCP 服务器会替您完成这些任务。参考 12.9 节了解有关 DHCP 的一般知识，还有配置我们举例的发行版本在系统引导时使用 DHCP 的特殊之处。

我们首先介绍配置的大致步骤，然后在针对各个厂商的小节里，回头说明每种发行版本的细节。其中用到的命令涉及了 Linux 内核的网络协议栈，它们在不同的发行版本上都是一样的。不过，每种发行版本为在系统引导时自动配置网络而建立了各自不同的配置文件，表 12.9 对这些文件做了总结。

表 12.9 网络配置文件

系 统	文 件	设 置 内 容
Red Hat，Fedora	/etc/sysconfig/network network-scripts/ifcfg-*ifname*	主机名、默认路由 IP 地址、网络掩码、广播地址
SUSE	/etc/rc.config /etc/route.conf	主机名、IP 地址、网络掩码等 默认路由
Debian，Ubuntu	/etc/hostname /etc/network/interfaces	主机名 IP 地址、网络掩码、默认路由

设计和安装一个物理网络的过程将在第 14 章介绍。如果您正管理着一个网络，并且对如何建网

www.TopSage.com

有一般性的概念，那么除非您要计划扩大现有的网络，否则可能没必要阅读太多关于物理连网方面的内容。

我们从以太网（Ethernet）的角度介绍网络配置的过程，其他连网技术本质上与之类似。

12.7.1 分配主机名和 IP 地址

关于怎样在本地网点最好地维护从主机名到 IP 地址的映射，管理员们有各种理论：通过 **hosts** 文件、NIS、DNS 系统或者可能是它们的某种组合。如果使用了多个系统，那么对于它们怎样共同运行必然还要有一个可行的计划。发生冲突的地方是要可扩展性和可维护性，还是要一个足够灵活、能让主机在所有服务还不齐全的时候启动（以及灵活到足以处理站点异构性）的网络系统。有关 DNS 的详细信息，请参见第 15 章。

另一个相关的长期性问题就是重新编号的问题。如果您的网络换了 ISP，您可能会不得不归还原来的 IP 地址，然后用从新 ISP 那里分到的地址重新编号。如果您必须访问网络上的每台主机来重新配置它，那么这个过程着实让人感到痛苦。为了加快重新编号的速度，可以在配置文件中使用主机名，让“主机名到 IP 地址”的转换只在 DNS 数据库中进行。另一方面，在配置文件中使用 IP 地址减少了在服务不齐全的时候引导系统所需要的依赖性。做与不做，好坏参半。

把名字映射到 IP 地址的最古老也最简单的方式就是**/etc/hosts** 文件。文件中的每一行以一个 IP 地址开头，后面跟着多个不同的符号名，即这个地址已知的名字。

下面是主机 lollipop 的**/etc/hosts** 文件：

```
127.0.0.1       localhost
192.108.21.48   lollipop.xor.com lollipop loghost
192.108.21.254  chimchim-gw.xor.com chimchim-gw
192.108.21.1    ns.xor.com ns
192.225.33.5    licenses.xor.com license-server
```

这个文件内容最少的版本只包含前两行。通常，**/etc/hosts** 文件把 localhost 作为第一行。

因为**/etc/hosts** 只包含本地映射关系，大多数现代系统都用它保存系统引导时所需的映射关系。然后查询 DNS，找到本地网络和外界的映射关系。您还可以用**/etc/hosts** 文件指定您不希望外界知道的映射关系，这些关系因而不会在 DNS 里公布。

/etc/hosts 文件曾经在启动过程中很重要，因为那时候还不能用 DNS，而且有时候在配置文件中用的是主机名而不是 IP 地址。现代 Linux 发行版本的确不需要**/etc/hosts** 文件，但是它可能还是应该至少包含主机本身和环回（loopback）地址的映射关系。加上默认网关主机和名字服务器的映射关系可能也会有帮助。许多站点都把它们的重要主机、服务器以及网关加到**/etc/hosts** 文件里，而别的一些站点则只加主机本身和环回接口的映射，还有一些站点会把所有的本地主机及其异地备份名字服务器的映射也加进去。

我们举例的 Linux 系统只安装了非常基本的**/etc/hosts** 文件。Debian 和 Ubuntu 的这个文件只包含 127.0.0.1 上的 localhost、127.0.1.1 上的系统主机名、以及一些 IPv6 信息。Red Hat 和 Fedora 的文件包含 localhost 和机器自己，而 SUSE 的包含 localhost 以及 localhost 和几个特殊 IPv6 名字的 IPv6 地址。

hostname 命令给一台机器分配一个主机名。**hostname** 命令一般在系统引导时从启动脚本中执行，它从配置文件中获得指定的名字。当然，每个厂商给这个配置文件起的名字都不一样。参考从 12.8 节开始的针对各厂商的内容，了解有关特定发行版本的信息。如今，大多数系统都会分配一个全名（也就是说，一个既包含主机名，又包含 DNS 域名的名字，比如 anchor.cs.colorado.edu）。

在比较小的网点上，您可以简单地手工分配主机名和 IP 地址。但是当牵扯到许多网络和许多不同的管理小组时，有某种中央协调机制来确保惟一性就很有帮助了。对于动态分配连网参数来说，DHCP

www.TopSage.com

会处理好惟一性的问题。有些站点现在使用 LDAP 数据库来管理他们的主机名和 IP 地址分配事务。参考 17.5 节了解有关 LDAP 的更多知识。

12.7.2 ifconfig：配置网络接口

ifconfig 能够启用或者禁用一个网络接口、设置它的 IP 地址和子网掩码、以及设置其他各种选项和参数。它常常在系统引导时运行（接受来自配置文件的命令行参数），但是也可以随时用来修改网络配置。请注意您是否在用 **ifconfig** 进行改动的同时又在从远程登录，许多系统管理员就是这样被关在了外面，只好驱车前往机房做纠正。

一条 **ifconfig** 命令最常见的形式是：

```
ifconfig interface address options ...
```

例如：

```
ifconfig eth0 192.168.1.13 netmask 255.255.255.0 up
```

参数 *interface* 标识这一命令所应用的硬件接口。在 UNIX 系统上，它通常是 2 或 3 个字符的设备名（取名源于网络接口卡所用的芯片组）后跟一个数字，但对于 Linux 来说，几乎总是像 eth0 这样的名字[17]。硬件的实际标识，以及到某个恰当的设备驱动程序的映射关系都保存在**/etc/modules.conf** 文件中的别名配置行中。环回接口称为 lo。

ifconfig *interface* 命令会显示出 *interface* 当前的设置，但并不会改变它们。许多系统都知道**-a** 意味着“所有接口”，所以 **ifconfig -a** 可以用来找出这个系统上的接口。如果您的系统不理解 **ifconfig -a**，那么可以试试 **netstat -i**，来找到接口名。

参数 *address* 指定了接口的 IP 地址。许多 **ifconfig** 的版本也接受一个主机名作为地址参数。我们宁愿使用实际的 IP 地址，如果给 **ifconfig** 一个主机名（或者是 **hostname** 命令的输出结果），那么在系统引导时会有更多的潜在问题出现。如果解析该主机名发生问题，那么这台主机就启动不起来了，或是会进入一种不能从网络访问的状态，这样就必须亲自到这台主机上解决这个问题。不能完成的 DNS 查询需要很长的时间才会超时退出，这使得主机看起来好像挂起了一样。另一方面，如果您曾经不得不对您的网络重新编号，那么在配置文件中找出所有直接写入的 IP 地址简直就是一场噩梦。

关键字 **up** 启动接口，**down** 则关闭它。当 **ifconfig** 命令给一个接口分配 IP 地址的时候，比如上面的那个例子，就暗含了参数 **up**，不需要再写它了。

ifconfig 命令还能理解许多其他选项。在这里我们只讨论最常见的几个，您可以查看手册页了解与您的系统有关的那些选项。所有的 **ifconfig** 选项都有符号名。有些选项需要一个参数，这个参数应该紧随选项名之后。

netmask 选项可设置接口的子网掩码，如果网络没有根据它的地址类别（A 类、B 类或者 C 类）划分子网，就需要这个选项。网络掩码能以十进制数的记法来指定，其中的各个部分用点（.）分开，也可以是以 **0x** 开头的 4 字节十六进制数字。在这两种情况下，设为 1 的比特位是网络号部分，而设置为 0 的比特位是主机号部分。

broadcast 选项指定接口的 IP 广播地址，它既可以用十六进制，也可以用以点分开的 4 字节记法表示。正确的广播地址的主机部分全设为 1，大多数系统都默认取这个值，它们用网络掩码和 IP 地址

[17] 您可以通过使用“虚拟网络接口”或者“IP 别名”这些概念，给一个网络接口分配多个 IP 地址。系统管理员经常用它们来让一台机器托管多个 Web 站点。在 Linux 系统上，虚拟接口称为 eth0:0、eth0:1，依此类推。您不需要事先声明这个接口是虚拟接口，只要用 **ifconfig** 命令把它们配起来就行了。参考 21.3 节了解更多信息。

www.TopSage.com

来计算广播地址。

在 Linux 上，可以把广播地址设置为任一 IP 地址，只要该 IP 地址在主机所连接的网络上有效即可。一些网点选择怪异的数值作为广播地址，借此希望避免某些基于广播 ping 的拒绝服务攻击。出于几种原因，我们不喜欢这种方法。

首先，它要求您重新设置本地网络的每台主机上的广播地址，对于一个大型网络而言，这是要花大量时间的工作。第二，它要求您绝对保证重新配置了各个主机，否则广播风暴就会导致包从一台计算机传送到另一台计算机，直到它们的 TTL 过期为止。

广播风暴产生的原因是，无论 IP 广播地址设置为什么，必须使用相同的链路层广播地址来传送包。例如，假设计算机 X 认为广播地址是 A1，计算机 Y 认为它是 A2。如果 X 发送一个包到地址 A1，那么 Y 会接收到这个包（因为链路层目的地址是广播地址），发现这个包并不是发送给它自己的，也不是给这个广播地址的（因为 Y 认为广播地址是 A2），这样它就会将这个包发回网络[18]。如果有两台计算机处在 Y 的状态下，那么包会一直循环传播下去，直到 TTL 过期。广播风暴会侵占带宽，尤其是在大型交换网络上更是如此。

为避免广播 ping 产生的问题，较好的一种方法是不让边界路由器转发它们，并告诉各个主机不要响应它们。请参考 12.11 节，了解如何实现这些约束条件。

在本节开头给出的 **ifconfig** 实例里，广播地址是 192.168.1.255，因为网络掩码 255.255.255.0 表明，这个网络是/24 网络。

执行 **ifconfig** 会显示如下输出：

```
redhat$ /sbin/ifconfig eth0
eth0  Link encap:Ethernet  HWaddr 00:02:B3:19:C8:86
      inet addr:192.168.1.13  Bcast:192.168.1.255  Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:206983 errors:0 dropped:0 overruns:0 frame:0
      TX packets:218292 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      Interrupt:7 Base address:0xef00
```

在这个例子中，Ethernet 接口上出现的冲突很少，这表明网络负载非常轻，或者更有可能，它是一个交换式网络。在共享式的网络（用集线器而不是交换机组建的网络）上，您应该检查这个数值，确保它在输出包数的大约 5%以下。大量的冲突表明网络负载很重，需要给予关注，可能的话应该划分为多个子网，或者迁移到交换式的网络结构上去。

现在让我们看几个完整的例子：

```
# ifconfig lo 127.0.0.1 up
```

这条命令配置了环回接口，环回接口一般不要求设置任何选项。您从不需要改变系统对这个接口的默认配置。隐含的网络掩码 255.0.0.0 是正确的，不需要手工覆盖它。

```
# ifconfig eth0 128.138.243.151 netmask 255.255.255.192
      broadcast 128.138.243.191 up
```

这是一个以太网接口的典型例子。IP 地址和广播地址分别被设置为 128.138.243.151 和 128.138.243.191。这个网络是 B 类网络（可以从地址的第一个字节辨别出来），但它已经用多加的 10 比特划分子网而成为一个/26 网络。网络掩码中的 192 用二进制数表示是 11000000，因此给包含 3 个 255 八位组的 24 比特位增加两位。广播地址里的 191 用二进制表示是 10111111，它把所有 6 个主机位

[18] 机器 Y 的配置必须启用了 ip_forwarding 功能才能出现这样的情况。

www.TopSage.com

都设置为 1，表明这个接口是按地址中第 4 个八位组划分出的 4 个网络里的第 3 个网络（前两位是 10）的一部分（这是体现出 IP 计算器程序方便性的一种情形）。

既然已经知道了怎样手工配置一个网络接口，现在就需要计算出在主机启动时怎样设置 **ifconfig** 的参数，同时还需要确保正确地输入这些新数值。通常是通过编辑一个或者多个配置文件做到这一点的。请看从 12.8 节开始的介绍各厂商的内容，了解详情。

12.7.3 mii-tool: 配置自动协商和其他介质特有的选项

偶尔，网络硬件会有一些可配置的选项是它们的介质所特有的。这类情况里极为常见的一个例子就是现代以太网，它的界面卡可以支持 10、100 乃至 1000Mbit/s 的半双工和全双工模式。大多数设备默认采取自动协商模式，网卡和它的上链连接（通常是一个交换机的端口）试着猜出对方想要用什么样的模式。

过去，自动协商过程往往和蒙着眼睛套小牛差不多。最近，制造商提供的网络设备总体表现好了很多，但是自动协商仍然是一个常出问题的地方。很高的丢包率（尤其是大包）是自动协商失败的常见结果。

避免这个缺陷的最好办法就是在服务器和它们所连接的交换机端口上都锁定接口的速率和工作模式（半双工或者全双工）。自动协商适用于公用区域中的端口，在这个区域内到处移动的笔记本电脑可能会停下来访问一下网络，但是对于静态连接的主机来说，自动协商就没什么用处了。如果碰到了原因未明的丢包问题，那么您应采取的第一步措施就是关闭各处的自动协商功能。

在 Linux 下，**mii-tool** 命令会查询并设置特定于介质的参数，比如链路速率和全/半双工模式。您可以用**-v** 标志查询接口的状态。例如，这里的 eth0 接口已经启动了自动协商功能：

```
$ mii-tool -v eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
  product info:  vendor 00:10:5a, model 0 rev 0
  basic mode:    autonegotiation enabled
  basic status:  autonegotiation complete, link ok
  capabilities:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:   100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
```

为了把这个接口锁定在 100Mbit/s 全双工的模式，可以使用命令：

```
# mii-tool -force=100BaseTx-FD eth0
```

把这条命令加到系统的启动脚本中，让它始终有效。然后查询状态，返回的结果为：

```
$ mii-tool -v eth0
eth0: 100 Mbit, full duplex, link ok
  product info:  vendor 00:10:5a, model 0 rev 0
  basic mode:    100 Mbit, full duplex
  basic status:  link ok
  capabilities:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:   100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
```

12.7.4 route：配置静态路由

route 命令定义了静态路由，静态路由是自始至终都明确不会改变（您希望如此）的路由表项，甚至当您运行了一个路由守护进程的时候也这样。在把一台新的计算机加到本地网络时，通常只需要

www.TopSage.com

大家网 TopSage.com

指定一条默认路由，参考下一节来了解详细内容。

本书对路由的讨论分为两部分，一部分是第 13 章，另一部分是本节。尽管这里给出了有关路由和 **route** 命令的大部分基本信息，但如果您需要了解更多的知识，看看第 13 章的前几节是很有帮助的。

路由是在 IP 层执行的。当要发送到其他某个主机的包到达的时候，就把这个包的目标 IP 地址和内核中路由表里的路由进行比较。如果它匹配或者部分匹配表中的某条路由，那么这个包就被转发到与这条路由相关联的“下一跳网关”的 IP 地址。

这里有两种特殊情况：首先，包可能被发送至一个直接连接的网络上的某台主机。在这种情况下，路由表里的“下一跳网关”的地址是本地主机自己的接口之一，所以包会被直接发送到它的目的地。在配置一个接口的时候可以用 **ifconfig** 命令将这种类型的路由添加到路由表中。

第二，可能没有与目的地址匹配的路由。在这种情况下，如果有默认路由的话就调用默认路由。否则，就会返回一个 ICMP“network unreachable（网络不可达）”或者“host unreachable（主机不可达）”消息给发送方。许多局域网只有一个出口，而且它们的默认路由指向这个出口。在 Internet 主干上，路由器没有默认路由——包就停在那里。如果它们没有某个目的地址的路由项，那么就不能到达这个目的地址。

每条 **route** 命令都增加或者删除一条路由。格式为：

```
route [op] [type] destination gw gateway [metric] [dev interface]
```

参数 *op* 应该是 **add**（增加路由）或者 **del**（删除路由），没有这个参数的话就显示路由表。*destination* 可以是一个主机地址（*type* 是**-host**）、一个网络地址（*type* 是**-net**）或者关键字 **default**。如果 *destination* 是一个网络地址，还应该指定一个网络掩码。

gateway 是应该转发包的那台主机。它必须处在一个直接连接的网络上，转发只能一次执行一跳。Linux 可以让您指定一个接口，代替（或者同时指定）*gateway*。在接口说明中的关键字 **dev** 是可有可无的，可以省略不用。

metric 是包到达目的地址所需要转发的次数（跳数）。Linux 不要求有或者使用跳数，但如果您设置了它，就在路由表中保留这个值，让路由协议可以用它。

可选的 *type* 参数用于支持主机路由，它需要一个完整的 IP 地址（某个特定的主机）而不是一个网络地址。*type* 参数接受**-net** 和**-host** 这两个值。如果没有指定 *type*，**route** 就会检查目的地址的主机部分是否为 0。如果主机部分为 0，或者地址是由**/etc/networks** 文件（如果您的系统有这个文件的话）定义的一个网络，那么这个路由就被当作一条普通的网络路由[19]。

由于 **route** 不知道哪个网络号已经划分了子网，所以您必须频繁地使用 *type* 域来安装特定路由。例如，地址 128.138.243.0 指的是我们的网点上划分了子网的 B 类网络，但对于 **route** 来说，它看起来好像是一个 B 类地址 128.138 带有一个主机部分 243.0，您必须指定**-net** 选项以消除 **route** 的混淆。一般情况下，较好的方法是为涉及子网的所有路由提供明确的 *type*。

route del *destination* 能从路由表中删除特定的项。其他 UNIX 系统的 **route** 命令有一个选项，通常是**-f** 或者**-flush**，它完全刷新路由表里的所有路由项，并且重新开始。Linux 不支持这个选项，所以您可能要面临用多条 **route del** 命令来清除一张很大的路由表的情况，此刻应确保您在本地登录系统，否则可能删到一半，您就被断开了。

若要检查现有路由，可使用 netstat -nr 命令，如果您想要看的是名字而不是数字的话，就使用 netstat -r 命令。因为域名查询可能会被打断，所以如果您正在做调试的话，最好还是用数字。

[19] **/etc/networks** 可以用于将名字映射至网络号，这与**/etc/hosts** 文件可将主机名映射为完整的 IP 地址非常相似。如果在**/etc/networks** 文件中（或者在 DNS 里）列出了网络名，则许多需要一个网络号的命令也可以接受一个网络名。

www.TopSage.com

```
redhat$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth0

redhat$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     *               255.255.255.0   U         0 0          0 eth0
127.0.0.0       *               255.0.0.0       U         0 0          0 lo
default         sprint-gw       0.0.0.0         UG        0 0          0 eth0
```

Genmask 是与目的地相关联的网络掩码。Flags 指出路由的状态，它是怎样学到的，以及其他参数。最后，Iface 是用这条路由将包发送出去的那个接口。这些例子取自于 Red Hat 系统，但是除了 Debian 默认不显示环回路由之外，SUSE 和 Debian 都和 Red Hat 是一样的。

12.7.5 默认路由

一条默认路由把所有在内核路由表里找不到目的网络的包发送到指定的网关。若要设置默认路由，只需简单地把下列代码行加到启动文件中：

```
route add default gw gateway-IP-address
```

大多数厂商都让它们的系统从配置文件获得网关 IP 地址，而不是把明确的 IP 地址直接写到启动文件。遗憾的是，对于我们举例的每一种 Linux 系统来说，本地路由信息加入到启动命令序列的方式各不相同（Linux 标准库[LSB]快点儿消除这种不该有的毛病吧！）。表 12.10 总结了我们举例的系统各自不同的方式。

表 12.10 如何设置默认路由

系 统	要修改的文件	要修改的变量
Red Hat，Fedora	/etc/sysconfig/network	GATEWAY
SUSE	/etc/route.conf	加入一行：default *IP-addr mask interface*
Debian，Ubuntu	/etc/network/interfaces	gateway

12.7.6 配置 DNS

要把一台主机配置为 DNS 客户机，只需编辑一、两个文件就行了：所有系统都要求修改 **/etc/resolv.conf**，有些系统还要求您修改“服务开关（service switch）”文件。

/etc/resolv.conf 文件列出了要解析不完全的名字（也就是说，不是全名，如 anchor，而不是 anchor.cs.colorado.edu）应该搜索的 DNS 域和名字服务器的 IP 地址以便查找名字。这里给出了一个实例，15.10.1 节里有更详细的介绍。

```
search cs.colorado.edu colorado.edu
nameserver 128.138.242.1
nameserver 128.138.243.151
nameserver 192.108.21.1
```

/etc/resolv.conf 应该首先列出“最近的”稳定的名字服务器，因为首先联络的是处在第一个位置

www.TopSage.com

的服务器。您最多可以有 3 个 nameserver 项。如果可能的话，应该始终有不止一项。向名字服务器做 DNS 查询的超时期似乎相当长，所以如果第一个名字服务器没有响应，您的用户会有感觉。

有时会看到一个 domain 行，而不是一个 search 行。之所以出现这样的一行，不是表明 **resolv.conf** 老文件尚未得到更新而不能使用 search 指令，就是表明老的解析器不能理解 search 指令。domain 规定了当前的名字域，而 search 接受多达 6 个不同的查询域，因此更应该用 search。

有些系统在默认条件下并不使用 DNS，甚至在有配置正确的 **resolv.conf** 文件存在的情况下也是这样。这些系统都有一个“服务开关（service switch）”文件，这个文件能确定用哪种机制来解析从主机到 IP 地址的映射。虽然在 17.4 节将详细地讨论信息资源的优先处理问题，但在这儿，我们也会提提这方面的内容，因为它有时会妨碍对一台新计算机的配置工作。参考第 17 章了解有关 NIS 的更多信息。

服务开关文件可让您指定参考 DNS、NIS 以及 **/etc/hosts** 的顺序。在大多数情况下，也可以完全取消某种数据源。选择的顺序会影响计算机的启动能力及启动过程与 **/etc/hosts** 文件中的内容交互的方式。

如果选择 DNS 作为第一个参考的数据源，那么可能需要在本地网上有一个名字服务器，并且在 **hosts** 文件里有它的主机名和 IP 地址，以便在启动时每样东西都能运行。

表 12.11 列出了我们举例的各个系统上相关配置文件的位置和主机查找的默认设置。

表 12.11 各个系统的服务开关文件

系 统	开 关 文 件	主机名查找的默认设置
Ubuntu	**/etc/nsswitch.conf** **/etc/host.conf**	files dns mdns[a] hosts, bind
其他系统	**/etc/nsswitch.conf** **/etc/host.conf**	files dns hosts, bind

a. mdns=muticast DNS（多播 DNS），一种不太常用的协议，没有本地 DNS 服务器也能在小型网络中解析类似 DNS 的名字。

12.7.7 Linux 连网协议栈

Linux 2.2 及以上版本内核中的连网协议栈都包括对虚拟网络接口、选择性确认（或者按它们的叫法 SACK）以及一种 IP 新特性——ECN（Explicit Congestion Notification，显式拥塞通知）的支持。ECN 通过标记 TCP 分组来通知发送方发生了拥塞，而不是丢弃分组作为问题的惟一指示。ECN 最初是在 RFC2481（1999 年 1 月）里规定的，现在则是 RFC3168 提出的建议性标准。RFC2884（2000 年 7 月）包含了 ECN 的性能数据和评测。人们发现 ECN 对于各种网络交易数据有好处。参考 21.3 节了解有关虚拟接口的更多知识。

Linux 总是吸收新特性的第一个网络协议栈之一。有时，Linux 用户使用新特性的速度快到让其他连网设施都不能与之互操作的程度。例如，Linux 的 ECN 特性（默认是启用的）会同 Cisco 防火墙产品上不正确的默认配置发生冲突，造成带有 ECN 位的全部分组都被丢弃。唉。

Linux 的开发人员热衷于修修补补，他们经常会实现尚未被接受成为标准的功能和算法。Linux 2.6.13 加上了可插拔的拥塞控制算法就是这样的一个例子。拥塞算法还有几种变化，用于有损的网络、有大量丢包的高速 WAN、卫星链路等。虽然默认仍然采用标准的 TCP “reno”机制（慢启动、避免拥塞、快速重传和快速恢复），但是某个算法变种或许更适合于您的网络环境。

12.8 特定于发行版本的网络配置

第 2 章详细介绍了我们举例的几种系统各自的引导过程。在接下来的几节里，我们简单总结一下和配置网络有关的若干任务。我们举例的系统都会自动配置环回接口，您绝对不需要修改那部分的配

www.TopSage.com

置。除此之外，每种系统略有不同。

对于我们举例的系统来说，有 4 个文件是它们都有的：**/etc/hosts**、**/etc/resolv.conf**、**/etc/nsswitch.conf** 和**/etc/host.conf**。这几个文件在前面介绍网络一般配置的章节里都讲过了。除了 **resolv.conf** 可能还有 **hosts**，在您把一台机器添加到网络中的时候，通常都不需要修改它们。

在对引导时刻能控制网络配置的文件做过任何修改之后，可能都需要重新启动一下系统，或者让网络接口先 down 再 up，以保证所做的改动起作用。在我们举例的所有发行版本上，您可以使用 **ifup** 和 **ifdown** 命令。

12.8.1 Red Hat 和 Fedora 的网络配置

表 12.12 给出了 Red Hat 和 Fedora 的网络配置文件。

表 12.12 Red Hat 和 Fedora 网络配置文件

在/etc/sysconfig 下的文件	设 置 内 容
network	主机名、默认路由
static-routes	静态路由
network-scripts/ifcfg-*ifname*[a]	每个接口的参数：IP 地址、网络掩码等

机器的主机名在文件**/etc/sysconfig/network** 中设置，这个文件还包括几行内容，指出这台机器的 DNS 域名和默认网关。例如，下面是一台机器上的 **network** 文件，它有一个以太网接口：

```
NETWORKING=yes
HOSTNAME=redhat.toadranch.com
DOMAINNAME=toadranch.com        ### optional
GATEWAY=192.168.1.254
```

特定于接口的数据保存在**/etc/sysconfig/network-scripts/ifcfg-*ifname*** 文件里，其中 *ifname* 是网络接口的名字。这几个配置文件能让您设置每个接口的 IP 地址、网络掩码、网络号和广播地址。它们还包括一行内容，指出这个接口是否应该在系统引导时“up”。

一般这个目录下会有用于以太网接口（eth0）的文件和用于环回接口（lo）的文件。例如：

```
DEVICE=eth0
IPADDR=192.168.1.13
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
```

和

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
```

是本节前面的 **network** 文件里提到的机器 redhat.toadranch.com 上的 **ifcfg-eth0** 和 **ifcfg-lo** 文件。

系统提供了两个便于进行接口管理的脚本。**ifup** 和 **ifdown** 接受一个网络接口名作为参数，并且启动（up）或者停止（down）指定的接口。改变了**/etc/sysconfig** 目录下任何文件中的网络信息之后，都要运行 **ifdown** *ifname* 命令然后再运行 ifup *ifname*。最好重新启动系统，以确保您所作的改动不会造

www.TopSage.com

成任何细微的问题。**ifup** 和 **ifdown** 没有手册页，但它们是 shell 脚本（保存在**/sbin** 下），所以您可以看看它们的内容，详细了解它们能做什么。

如果您需要一次管理所有的接口，那么可以运行脚本**/etc/rc.d/init.d/network**，这个脚本接受的参数有 **start**、**stop**、**restart** 和 **status**。在系统引导时，是以 **start** 参数调用这个脚本的。

系统的启动脚本还可以配置静态路由。在系统引导时刻，添加到文件**/etc/sysconfig/static-routes** 里的所有路由都进入到路由表中。这个文件里的配置项给 **route add** 提供了参数，虽然顺序是乱的（先给而不是最后给接口）。

```
eth0 net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49
eth1 net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129
```

首先指定接口，然后是路由参数：路由类型（net 或者 host）、目标网络、与这个网络相关联的网络掩码，最后是下一跳网关。需要用关键字 gw。目前的 Linux 内核虽然不用参数 metric，但允许引入并在路由表中保留这个参数，供路由守护进程使用。上面静态路由的例子会得到下面的 **route** 命令：

```
route add-net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49 eth0
route add-net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129 eth1
```

12.8.2 SUSE 的网络配置

表 12.13 给出 SUSE 用于网络配置的文件。

表 12.13 SUSE 在/etc/sysconfig/network 下的网络配置文件

文 件	设 置 内 容
ifcfg-*interface*	主机名、IP 地址、网络掩码等
ifroute-*interface*	特定于接口的路由规定
routes	所有接口的默认路由和静态路由
config	许多不常用的网络变量

对于一个真实的网络接口（也就是说，不是环回接口）来说，文件名采用扩展形式 **ifcfg-***interface*-**id-***MAC*，其中 *MAC* 是网络接口的硬件地址（例如 ifcfg-eth-id-00:0c:29:d4:ea:26）。

除了指定一个接口的 IP 地址、网关和广播信息之外，**ifcfg-***文件能调节许多别的网络设置，**ifcfg.template** 文件中很好地总结了可能有的参数。

SUSE 的 YaST 工具带有一个配置网络的接口文件模板。它效果不错，我们建议尽可能用它管理 **ifcfg-***文件。如果您必须手工配置网络，下面是一个简单的模板，并附有我们的注释：

```
BOOTPROTO='static' # 隐含为静态（static），但多说明一下也无妨
IPADDR='192.168.1.4/24' # /24 规定了 NETWORK 和 NETMASK 变量
NAME='AMD PCnet - Fast 79C971' # 用于启动和停止接口
STARTMODE='auto' # 在系统引导时自动启动
USERCONTROL='no' # 禁止通过 kinternet/cinternet GUI 的控制
```

SUSE 系统的静态全局路由信息（包括默认路由）保存在文件 **routes** 里。这个文件的每一行看上去都像是一条省略了命令名的 route 命令：目的地址、网关、网络掩码、接口以及可以保存在路由表里供路由选择守护进程使用的其他可选参数。对于上面配置的主机而言，它只有一条默认路由，这个 **routes** 文件包含的内容为：

www.TopSage.com

```
default 192.168.1.1 - -
```

只针对特定接口的路由保存在 ifroute-interface 文件里，这里 interface 的命名方法和 ifcfg-*文件名的接口部分命名方法一样。文件的内容采用和 routes 文件一样的格式。

12.8.3 Debian 和 Ubuntu 的网络配置

如表 12.14 所示，Debian 和 Ubuntu 的网络配置大多出现在文件**/etc/hostname** 和文件**/etc/network/interfaces** 里，另外从**/etc/network** 目录下的 **options** 文件里提供一点儿帮助。

主机名在**/etc/hostname** 文件里设置。这个文件里的名字应该是全名，因为它的值要在各种场合下使用，其中有些场合需要全名。不过，标准的 Debian 安装在那里留下的不是一个全名。

表 12.14 Debian 和 Ubuntu 的网络配置文件

文 件	设 置 内 容
/etc/hostname	主机名
/etc/network/interfaces	IP 地址、网络掩码、默认路由
/etc/network/options	下层网络选项（IP 转发等）

IP 地址、网络掩码和默认网关在文件**/etc/network/interfaces** 里设置。每个接口有以关键字 iface 开头的一行，后面跟着多个缩行，指定更多的参数，例如：

```
iface lo inet loopback
iface eth0 inet static
    address 192.168.1.102
    netmask 255.255.255.0
    gateway 192.168.1.254
```

ifup 和 **ifdown** 命令读取这个文件，通过正确的参数调用下层的命令，比如 **ifconfig**，启动或者停止接口。`iface` 行中的关键字 `inet` 是地址族，它一定是 `inet`。关键字 static 称为一种“方法”，它表示直接给 eth0 分配 IP 地址和网络掩码。静态配置要求有 `address` 和 `netmask` 行，较早的 Linux 内核版本还要求指定网络地址，但是现在的内核更聪明了，它能从 IP 地址和网络掩码推算出网络地址来。`gateway` 行指定默认网关的地址，用它来安装一条默认路由。

options 文件能够在系统引导时设置某些连网变量。在默认情况下，Debian 设置关闭 IP 转发（IP forwarding）功能、打开欺骗保护（spoof protection）、关闭 cookie 同步（syn cookies）功能。

12.9 DHCP：动态主机配置协议

Linux 主机过去需要手工配置才能加入到一个网络中。可您只要把 Mac 或者 PC 插上网络，网就通了。为什么 Linux 不能这样做呢？DHCP（动态主机配置协议，Dynamic Host Configuration Protocol）让这个合理的愿望向现实迈近了几步。DHCP 在 RFC2131 和 2132 里定义。

这个协议可以让一台 DHCP 客户机从中央服务器“租用”各种网络和管理参数，中央服务器得到了授权，可以分配这些参数。对于不使用时就关闭的 PC 和拥有间歇性拨号上网客户的 ISP 而言，这种租用模式尤其方便。

可租用的参数包括：

- IP 地址和网络掩码；

www.TopSage.com

- 网关（默认路由）；
- DNS 名字服务器；
- syslog（系统日志）主机；
- WINS 服务器、X 字体服务器、代理服务器、NTP 服务器；
- TFTP 服务器（用于加载一个启动图像）。

以及其他数十种参数（参考 RFC2132）。不过，在现实世界中很少使用更特殊的参数。在许多情况下，一台 DHCP 服务器只提供基本的网络参数，如 IP 地址、网络掩码、默认网关和名字服务器。

客户机必须定期向 DHCP 服务器报告，为租用的参数续约。如果没有续约，最后就会过期。到时 DHCP 就可以自由地将地址（或者任何可以租用的参数）分配给其他客户机了。租用期是可以配置的，但通常是相当长的时间（数小时或数天）。

DHCP 可以为从前倒霉的系统管理员节省大量的时间，减少他们的苦恼。打开服务器并且它开始运行之后，客户机在启动时就可以利用它自动得到它们的网络配置了。不慌不忙、有条不紊。

12.9.1 DHCP 软件

Linux 发行版本以前曾带有各种各样不同的 DHCP 服务器和客户机软件。现如今，它们都多少形成了标准，采用 ISC（Internet 软件联盟，Internet Software Consortium）的参考实现。ISC 服务器也支持 BOOTP 协议，该协议在概念上与 DHCP 很相似，但是要古老一些，也要简单一些。在现代所有的 Linux 发行版本上，都默认安装有 DHCP 客户机软件，但是您有时候必须安装更多软件才能让 ISC 服务器和中继代理启动和运行。

DHCP 客户机必须采用通用的全 1 广播地址发起与 DHCP 服务器的对话，因为客户机还不知道它们的子网掩码，所以不能使用子网广播地址。

ISC 的 DHCP 服务器使用 DNS 动态更新协议。这个服务器不仅为主机分配 IP 地址和其他连网参数，而且还能用正确的主机名到 IP 地址的映射更新 DNS 数据库。请参考 15.14 节有关动态 DNS 更新的内容。

在接下来的几节里，我们先简要讨论 DHCP 协议，说明如何设置实现 DHCP 协议的 ISC 服务器，然后再讨论 DHCP 客户机配置问题。

12.9.2 DHCP 的工作方式

DHCP 是对 BOOTP 的扩展，它向后兼容 BOOTP，BOOTP 最初的设计目的是为了引导 UNIX 无盘工作站。BOOTP 为客户机提供了 IP 地址、网络掩码、默认网关以及 TFTP 引导信息。DHCP 总结归纳了可以提供的参数，并加上“租用”概念。

一台 DHCP 客户机是通过广播一则消息“Help！Who am I?（救命！我是谁？）”而开始与 DHCP 服务器进行交互的。如果本地网络上有一台 DHCP 服务器，它就和客户机进行协商，把 IP 地址租给客户机，并提供其他连网参数（网络掩码、名字服务器信息和默认网关）。如果本地网络上没有 DHCP 服务器，那么位于不同子网的服务器也能从叫做“relay agent（中继代理）”的代理那里接收到一开始的广播消息。

在客户机的租期过半的时候，它会续约。服务器必须跟踪它已经分出去的地址，而且在系统重新启动之后也必须保留这些信息。客户机也应该在重新启动之后保留它们的租借状态，但许多客户机做不到这一点。这样做的目的是为了使网络的配置最为稳定。

顺便提一句，通常 DHCP 不用于配置拨号 PPP 接口。一般由 PPP 自己的 PPPCP（PPP Control Protocol，PPP 控制协议）完成这项任务。

www.TopSage.com

12.9.3 ISC 的 DHCP 服务器

为了配置 DHCP 服务器 **dhcpd**，需要编辑来自 **server** 目录的 **dhcpd.conf** 样本文件，并把它安装为/etc/dhcpd.conf[20]。同时您还必须创建一个叫做**/var/db/dhcp.leases** 的空的租用数据库文件，用 **touch** 命令就行。确保 **dhcpd** 可以写这个文件。为了设置 **dhcpd.conf** 文件，需要下面的信息：

- 由 **dhcpd** 管理 IP 地址的子网，以及可以发放的地址范围；
- 按秒计算的初始和最大的租期；
- BOOTP 客户机的配置，如果有的话（它们具有静态 IP 地址，同时必须列出它们的 MAC 级硬件地址）；
- 服务器应该传送给 DHCP 客户机的其他任何选项：网络掩码、默认路由、DNS 域、名字服务器等。

dhcpd 手册页对配置过程进了概述。配置文件的确切语未能在 **dhcpd.conf** 手册页中予以介绍。两者都位于软件的 **server** 子目录中。有些 Linux 发行版本的**/etc** 目录下包含了一个 **dhcpd.conf** 文件的样板；您可以修改它，使之符合您本地站点的配置。

dhcpd 应该在系统引导时刻自动启动。您会发现，根据**/etc/dhcpd.conf** 文件的存在与否而有条件地启动守护进程的做法很有帮助。

下面是从一个有两个接口的 Linux 主机得到的 **dhcpd.conf** 文件样本：其中一个接口是内部接口，另一个接口与 Internet 连接。这台计算机为内部网络执行 NAT 转换，同时也在这个网络上出租 10 个 IP 地址。**dhcpd.conf** 文件包括用于外部接口的一个哑项（必需的）和为每台需要一个固定地址的特定计算机提供的一个 host 项。

```
# dhcpd.conf
#
# global options
option domain-name "synack.net";
option domain-name-servers gw.synack.net;
option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.51 192.168.1.60;
    option broadcast-address 192.168.1.255;
    option routers gw.synack.net;
}

subnet 209.180.251.0 netmask 255.255.255.0 {
}

host gandalf {
    hardware ethernet 08:00:07:12:34:56;
    fixed-address gandalf.synack.net;
}
```

DHCP 分配的地址可能与 DNS 数据库的内容冲突。网点经常给每个动态租用地址分配一个通用名（如 dhcpl.synack.net），并且允许各个机器名在它们的 IP 地址之间“浮动”。如果您运行的是最近发行的支持动态更新的 BIND 版本，那么也可以配置 **dhcpd**，在它分配地址的时候更新 DNS 数据库。动态更新解决方案更为复杂，但它有一个优点，可以保存每台计算机的主机名。参考第 15 章了解有关

[20] 请注意：**dhcpd.conf** 文件格式比较脆弱。如果遗漏一个分号，就会收到一个含糊而无用的错误信息。

www.TopSage.com

DNS 的更多信息。

dhcpd 在 **dhcp.leases** 文件里记录各项出租事务。它也会通过将这个文件重命名为 **dhcpd.leases~**，再从它在内存中的数据库重新创建 **dhcp.leases** 文件这样的方法定期备份这个文件。如果 **dhcpd** 在这个操作过程中崩溃了，那么可能最后只留下一个 **dhcp.leases~**文件。在这种情况下，**dhcpd** 将拒绝启动，您必须在重新启动之前对这个文件重新命名。不要只建立一个空的 **dhcp.leases** 文件，否则当客户机有两个地址的时候就会产生混乱。

DHCP 客户机实际上不需要配置。它把每个连接的状态文件保存在目录**/var/lib/dhcp** 或者**/var/lib/dhclient** 下。这些文件的名字是用它们所描述的接口来命名的。例如，**dhclient-eth0.leases** 文件里包含 **dhclient** 已经给接口 eth0 设置的所有连网参数。

12.10 Linux 的动态重新配置和调整

Linux 有它自己调整内核和连网参数的特殊方式。Linux 没有提供一个普通配置文件供读取来设置这些参数，而是把能调整的每个变量的表示都放在虚拟文件系统**/proc** 下。连网变量都在目录**/proc/sys/net/ipv4** 下：

```
$ cd /proc/sys/net/ipv4; ls -F
conf/                                ip_local_port_range     tcp_mem
icmp_echo_ignore_all                 ip_nonlocal_bind        tcp_moderate_rcvbuf
icmp_echo_ignore_broadcasts          ip_no_pmtu_disc         tcp_no_metrics_save
icmp_errors_use_inbound_ifaddr       neigh/                  tcp_orphan_retries
icmp_ignore_bogus_error_responses    route/                  tcp_reordering
icmp_ratelimit                       tcp_abc                 tcp_retrans_collapse
icmp_ratemask                        tcp_abort_on_overflow   tcp_retries1
igmp_max_memberships                 tcp_adv_win_scale       tcp_retries2
igmp_max_msf                         tcp_app_win             tcp_rfc1337
inet_peer_gc_maxtime                 tcp_congestion_control  tcp_rmem
inet_peer_gc_mintime                 tcp_dma_copybreak       tcp_sack
inet_peer_maxttl                     tcp_dsack               tcp_stdurg
inet_peer_minttl                     tcp_ecn                 tcp_synack_retries
inet_peer_threshold                  tcp_fack                tcp_syncookies
ip_autoconfig                        tcp_fin_timeout         tcp_syn_retries
ip_default_ttl                       tcp_frto                tcp_timestamps
ip_dynaddr                           tcp_keepalive_intvl     tcp_tso_win_divisor
ip_forward                           tcp_keepalive_probes    tcp_tw_recycle
ipfrag_high_thresh                   tcp_keepalive_time      tcp_tw_reuse
ipfrag_low_thresh                    tcp_low_latency         tcp_window_scaling
ipfrag_max_dist                      tcp_max_orphans         tcp_wmem
ipfrag_secret_interval               tcp_max_syn_backlog
ipfrag_time                          tcp_max_tw_buckets
```

名字里带有 **rate** 和 **max** 的变量有许多都可以防止发生 DoS（denial of service，拒绝服务）攻击。**conf** 子目录包含的变量都是按接口来设置的。它包含子目录 **all** 和 **default**，还有每个接口的一个子目录（包括环回接口）。每个子目录都包含一组相同的文件。

```
$ cd conf/default; ls -F
accept_redirects      bootp_relay          log_martiansrp_filter
accept_source_route   disable_policy       mc_forwardingsecure_redirects
arp_announce          disable_xfrm         medium_idsend_redirects
arp_filter            force_igmp_version   promote_secondariesshared_media
arp_ignore            forwarding           proxy_arptag
```

www.TopSage.com

如果您改动了子目录 **all** 里的什么地方，那么您的改动会施加到所有的接口上。如果您修改了，比如说子目录 **eth0** 里的同一个变量，那么只有这个接口才受影响。**defaults** 子目录包含原来带有的默认值。

neigh 目录里也为每个接口包含一个子目录。每个子目录的文件为那个特定接口管理 ARP 表以及发现 IPv6 邻居。下面列出了若干变量，以 **gc** 开头的那些变量（代表 garbage collection）决定了 ARP 表项是怎样超期和被丢弃的。

```
$ cd neigh/default; ls -F
anycast_delay              gc_interval      locktimeretrans_time_ms
app_solicit                gc_stale_time    mcast_solicitucast_solicit
base_reachable_time        gc_thresh1       proxy_delayunres_qlen
base_reachable_time_ms     gc_thresh2       proxy_qlen
delay_first_probe_time     gc_thresh3       retrans_time
```

用 **cat** 可以看到一个变量的值，用重定向到适当文件名的 **echo** 命令可以设置它的值。例如，命令：

```
$ cat icmp_echo_ignore_broadcasts
0
```

显示这个变量的值是 0，这意味着不会忽略广播 ping。要把它设置为 1（从而避免遭受 smurf 类的拒绝服务攻击），可以在**/proc/sys/net** 下运行：

```
$ sudo sh -c "echo 1 > icmp_echo_ignore_broadcasts"[21]
```

在您调整这些变量的时候，您一般是通过您正在调整的网络登录进来的，所以要小心！您可能会把事情搞糟，结果不得不从控制台重新启动才能恢复，如果系统碰巧在是阿拉斯加的 Point Barrow，时间又是 1 月份，那可就麻烦了。在您考虑在一台工作主机上动手之前，先在台式机上试着调整一下这些变量。

若要永久改变这些参数的任何一个（或者更准确地说，在系统每次启动的时候再次重置它们），就要把正确的变量加到**/etc/sysctl.conf** 文件里，在系统引导时 **sysctl** 命令会读这个文件。**sysctl.conf** 文件的格式为 *variable=value*，而不是您在 shell 里手工修改这个变量时要运行的命令 **echo value > variable**。变量名是相对于**/proc/sys** 的路径名，如果愿意，您还可以使用圆点（.）来代替斜线（/）。例如，**/etc/sysctl.conf** 文件里有下面两行中的任何一行都会关闭（这台主机的）IP 转发功能。

```
net.ipv4.ip_forward=0
net/ipv4/ip_forward=0
```

/usr/src/linux/Documentation/proc.txt 这个文档是由 SUSE 的爱好者写的，它是有关用**/proc** 调整内核不错的入门教材[22]。它告诉您各个变量的真实含义是什么，有时还给出了建议值。文件 **proc.txt** 稍微有点儿过时，因为写 Linux 代码的人似乎比写这个文档的人速度快一些。

12.11 安全问题

我们将在第 20 章专门讨论安全问题，但在这里值得讨论一些与 IP 连网有关的安全问题。在这一

[21] 如果您试着用以 **sudo echo 1 > icmp_echo_ignore_broadcasts** 的形式运行这条命令，只会产生一条“permission denied”消息——您的 shell 试图在运行 **sudo** 之前打开输出文件。您要让 **sudo** 既用于 **echo** 命令，也用于重定向。因此，您必须创建一个有 root 权限的子 shell，然后在其中执行整条命令。

[22] 您必须安装内核源代码，才会有 **proc.txt** 这个文件。

www.TopSage.com

节，我们会简单地看一下几个因为能引起安全问题而知名的连网特性，而且推荐几种可以将它们的影响减到最小的方法。我们举例的 Linux 系统对这些问题的默认处理方式（以及改正它们所采取的正确方法）将在这一节的后面详细介绍。

12.11.1 IP 转发

如果一台 Linux 主机启用了 IP 转发（IP forwarding）功能，那么它就成了一台路由器。除非系统具有多个网络接口，而且确实要求作为一台路由器运行，否则我们建议您关闭这个功能。转发包的主机有时会因为让外来的包看起来像是来自网络内部而造成安全漏洞。这种诡计可以帮助“调皮捣蛋的”包逃避网络扫描程序和包过滤程序。

12.11.2 ICMP 重定向

ICMP 重定向可以被恶意使用来重新路由流量，造成您的路由表混乱。大多数操作系统在默认条件下都会听它们的话，遵循它们的指示。如果所有的流量被重新路由至一个竞争对手的网络达几个小时，尤其是同时备份线路还在运行的话（译者注：如果备份线路运行，那么流量会从备份线路通过，不容易发现默认线路路由不正常），情况就非常糟糕了。建议您配置路由器（以及作为路由器的主机）忽略 ICMP 重定向，可能的话还可以做 ICMP 重定向的日志记录。

12.11.3 源路由

IP 的源路由机制可以让您为发送至目的地址的包指定明确的一系列网关。源路由绕过了下一跳路由算法，这个算法通常在每个网关上运行，用以确定如何转发包。

源路由是最初的 IP 规范的组成部分，它的目的主要是协助测试。它可以造成安全问题，因为包常常是根据它们的源进行过滤的。如果某人能够狡猾地路由一个包，使它看起来像是来自于自己网络的内部而不是来自 Internet，那么它就有可能逃过您的防火墙。我们建议您既不要接收源路由包，也不要发送源路由包。

12.11.4 广播 ping 和其他形式的定向广播

地址是某个网络的广播地址（而不是某个特定的主机地址）的 ping 包一般将发送到这个网络上的所有主机。这样的包常用来进行 DoS 攻击（Denial of Service，拒绝服务），例如，所谓的 smurf 攻击。大多数主机都有办法禁用广播 ping——也就是说，可以配置它们不响应或者转发这种广播 ping。Internet 路由器也可以在广播 ping 到达内部网络之前把它们过滤掉。如果可以的话，同时采用主机和防火墙级的安全措施不失为一种好方法。

广播 ping 是“定向广播”的一种形式，因为它们是被发送到一个远程网络的广播地址的包。对这种包的默认处理不断地发生着变化。例如，Cisco IOS 在 11.x 之前的版本默认条件下都会转发定向的广播包。但 12.0 以后的版本就不能转发这样的包。一般情况下可以让 TCP/IP 协议栈忽视来自远程网络的广播包，但由于这种行为必须在所有接口上设置，所以在大型网点上这并非易事。

12.11.5 IP 欺骗

IP 包的源地址在正常情况下是由内核的 TCP/IP 实现填写的，它是发送包的主机的 IP 地址。不过，

www.TopSage.com

如果创建包的软件使用了 raw 类型的套接口，那么它就可以填进去它喜欢的任何源地址。这叫做 IP 欺骗（IP spoofing），它常常和某种类型的网络恶意行为有关联。由伪造的源 IP 地址（如果它是一个真实的 IP 地址的话）所标识出的机器，往往是这种恶意行为的牺牲品。出错而返回的包会中断或者阻塞这个牺牲品的网络连接。

您应该在边界路由器上封锁住任何源地址不在您的地址空间中的包，以此拒绝 IP 欺骗的攻击。如果您的网络是一所大学，学生们喜欢做试验，并且会对他们喜爱的聊天频道上的“混蛋们”有报复心，事先这样做就特别重要了。

同时，如果您在内部使用了私用地址空间，那么可以过滤任何漏到 Internet 上的内部地址。这样的包肯定得不到应答（因为在主干上没有路由），而且往往表明您的网点内部配置有错误。

下一节会介绍基于 Linux 的防火墙，它们提供了一种基于主机的过滤机制。不过，大多数网点都更愿意在它们的边界路由器而不是在每台主机上实现这类过滤机制。这也是我们建议采用的方法。我们在这里介绍基于主机的防火墙，只是为了让内容完整，并说明它的特殊应用场合。

您也必须提供保护，防止因为黑客伪造外来包的源地址，让您的防火墙误以为它们源自于您的内部网络。内核参数 **rp_filter**（位于目录**/proc/sys/net/ipv4/conf/***ifname* 下）能够帮助您检测到这样的包，**rp** 代表反向路径（reverse path）。您把这个变量设为 1，如果某个接口上收到的包和把它的源地址作为目的地址从这个接口上发出去的包不一样，那么内核就会丢弃它。在默认情况下，这种功能是打开的。

如果您的网点有多条到 Internet 的连接，那么入和出的路由完全有理由不一样。在这种情形下，您必须把 **rp_filter** 设为 0，才能让您的路由协议正常工作。如果您的网点只有一条通往外部 Internet 的连接，那么把 **rp_filter** 设为 1，通常比较安全和合理。

12.11.6 基于主机的防火墙

Linux 包含了包过滤（packet filtering，也叫做“防火墙”）软件。虽然我们随后将在这一章（12.12 节）以及第 20 章（20.13 节）里介绍这个软件，但我们确实不推荐用户把一台工作站用作防火墙。Linux 的安全性（特别是我们善意的厂商提供的安全性）很薄弱，不过 Windows 的安全性更差。我们建议您购买一种专用的硬件解决方案，用作防火墙。哪怕像 Check Point 公司的产品 FireWall-1（它在 Solaris 主机上运行）那样高级的软件解决方案都没有像 Cisco 的 PIX 那样的专用硬件好，它们的价钱几乎是一样的！

从 20.12 节开始，有关于防火墙问题更全面透彻的讨论。

12.11.7 虚拟私用网络

许多机构的办事处分布在世界上的不同地方，他们希望让所有办事处都连接到一个大型的私用网络上。遗憾的是，租用跨洋乃至跨洲数据线路的费用高得惊人。这样的机构实际上可以通过在其各个不同的办公地点之间建立一系列安全、加密的“隧道”，把 Internet 当作就好像一条私用数据线路那样来用。包含这类隧道的“私用”网络就叫做虚拟私用网络（virtual private network）或者 VPN。

有些 VPN 采用 IPsec 协议，该协议在 1998 年成为 IETF 的标准。别的 VPN 使用专有的解决方案，彼此之间通常没有互操作性。如果您需要 VPN 功能，我们建议您考察一下像 Cisco 3660 路由器或者 Watchguard Firebox 这样的产品，两者都能实现隧道和加密功能。Watchguard 公司的设备用 PPP 连接到串口来进行管理。系统管理员可以拨号进入设备来配置它或者访问 VPN 进行测试。参考 20.14 节了解有关 IPsec 的更多知识。

对于低预算的 VPN 解决方案，可以参考 12.13 节里的例子，它在 **ssh** 连接上使用 PPP 来实现虚拟私用网络。

www.TopSage.com

12.11.8 与安全有关的内核变量

表 12.15 列出了 Linux 对于各种棘手的网络问题默认采取的处理方式。本节简要介绍了这些处理方式所暗含的意思。我们建议您修改这些变量的值，不要应答广播 ping、不要听路由重定向的指挥、也不要接受任何源路由包。

表 12.15 Linux 中与安全有关的默认网络行为

功 能	主 机	网 关	控制文件（在/proc/sys/net 下）
IP 转发	关闭	打开	**ipv4/ip_forward** 用于整个系统 **ipv4/conf/*interface*/forwarding** 用于每个接口[a]
ICMP 重定向	服从	忽略	**ipv4/conf/*interface*/accept_redirects**
源路由	忽略	服从	**ipv4/conf/*interface*/accept_source_route**
广播 ping	应答	应答	**ipv4/icmp_echo_ignore_broadcasts**

a. 这个 *interface* 既可以是一个特定的接口名称，也可以是 **all**（所有的接口）。

12.12 Linux 的 NAT

传统的 Linux 只实现了一种有限形式的 NAT（Network Address Translation，网络地址转换），称之为 PAT（Port Address Translation，端口地址转换）更合适一些。PAT 没有像真正的 NAT 实现那样使用一定的 IP 地址范围，而是将多条连接复用到一个地址上。更混乱的是，许多 Linux 文档常常把这项功能既不叫 NAT，也不叫 PAT，而是叫做“IP 伪装（IP masquerading）”。其间的细节和差异没有太大的实际意义，所以为了保持一致性，我们把 Linux 的实现就叫做 NAT。

iptables 不仅实现了 NAT，而且还实现了包过滤机制。在 Linux 的早期版本里，这有点儿混乱，但 **iptables** 将 NAT 和过滤功能更清楚地划分开来了。

包过滤功能将在讲安全的章节里（20.12 节）详细介绍。如果您使用 NAT 让本地主机访问 Internet，那么在运行 NAT 的时候，必须使用完整的防火墙过滤器予以配合。NAT“不是真正的 IP 路由选择”，这一事实让 Linux 的 NAT 网关不会比 Linux 路由器更安全。为简洁起见，我们在这儿仅限于介绍实际的 NAT 配置，不过，NAT 配置只是完整配置的一小部分。

若要 NAT 发挥作用，您必须通过把内核变量/proc/sys/net/ipv4/ip_forward 设置为 1，启动内核里的 IP 转发功能。此外，您还必须插入适当的内核模块：

```
$ sudo /sbin/modprobe iptable_nat
$ sudo /sbin/modprobe ip_conntrack
$ sudo /sbin/modprobe ip_conntrack_ftp
```

使用 NAT 路由分组的 **iptables** 命令形式如下：

```
$ sudo iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 63.173.189.1
```

在本例中，eth0 是连接到 Internet 的接口，它的 IP 地址是--to 的参数。接口 eth1 是连接内部网络的接口。

对于 Internet 主机来说，从内部网络的主机发来的所有包都带有 eth0 的 IP 地址。执行 NAT 功能的主机接收传入的包，检查它们的真实目的地址，用正确的内部网络 IP 地址重写这个目的地址，然后再把它们送上愉快的行程。

www.TopSage.com

12.13 PPP：点对点协议

PPP 即点对点协议（Point-to-Point Protocol），它以既可以在最慢的 Internet 链路上使用，又可以在最快的 Internet 链路上使用而著称。同步方式的 PPP 是一种封装协议，可以用在链路两端有胖路由器（fat router）的高速电路上。异步方式的 PPP 是一种串行线路封装协议，它指定了必须如何对 IP 包进行编码好在一个慢速的（通常是不可靠的）串行线路上传输。串行线路只是简单地发送位流，它对一个包的起始或结束一无所知。PPP 设备驱动程序负责串行线路上的包编码和解码，它加入了一个链路层包头以及分隔包的标记。

有时 PPP 与 DSL 和线缆调制调解器（cable modem）这样的新兴家用技术一同使用，但作为一个系统管理员常常看不到这一点。一般情况下，接口设备执行封装，而流量被桥接至以太网。这样一来，您就只会看到一条以太网连接。

PPP 由专家委员会设计，它是能“包罗万象”的封装协议。它的灵感来自于由 Rick Adams 和 Van Jacobson 分别设计的 SLIP（Serial Line IP，串行线路 IP）和 CSLIP（压缩的串行线路 IP）。PPP 与这些系统不同，因为它允许通过一条链路传输多种协议。它的规范在 RFC1331 中规定。

12.13.1 解决 PPP 的性能问题

PPP 提供了以太网的所有功能，但速度要慢得多。一般办公室里的 LAN 运行速度是 100Mbit/s 或者 1Gbit/s——即 100 000~1 000 000kbit/s。拨号上网的速度大约为 28~56kbit/s。对这些数字直观些的说法是，通过一条 PPP 线路传送一个一兆字节的文件大概需要 5 分钟。对于电子邮件或者关闭图像进行 Web 浏览来说，这个速度还可以，但对于漂亮的 Web 站点来说，这个速度会慢得让您发疯。为了提高交互的性能，您可以将点对点的 MTU 设置得很低。它的默认值通常是 512 字节，如果您在做许多交互性的工作，可以试试 128 字节。如果您在用 PPPoE（PPP over Ethernet），那么可以用 **tcpdump** 查看一下网络上通过的包的大小，并相应地设置 MTU。以太网的 MTU 是 1500，但是 PPP 封装使得稍小一点儿的值效率更高。例如，**pppoe** 建议 PPP 连接后面的主机为 1412 字节，而建议 PPP 链路为 1492 字节。您一定不想因为设置的默认 MTU 太大而让每个包都被分片。

在 PPP 链路之上运行 NFS 慢得让人难受。只应该在能够以 TCP 而不是 UDP 运行 NFS 的时候考虑这种做法。有关 NFS 的更多信息，请参考第 16 章。

X Windows 系统协议使用 TCP，所以通过一条 PPP 链路运行 X 应用是可能的。像 **xterm** 这样的程序运行情况良好，但应避免使用花哨的字体或位图图形的应用程序。

12.13.2 用 PPP 连网

若要用 PPP 把主机连接到网络上，必须满足下面 3 个前提条件：

- 主机的内核必须能够按 PPP 协议标准指定的那样通过一条串行线路传送 IP 包；
- 必须有一个允许您建立和维护 PPP 连接的用户级程序；
- 在串行线路的另一端必须有一个能理解您所使用的协议的主机。

12.13.3 让主机使用 PPP

若要建立 PPP 连接，主机必须能够发送和接收 PPP 包。在 Linux 系统上，PPP 是一个可加载的内核模块，它将网络包放在串行设备的输出队列中，反之也可以从输入队列接收网络包。这个模块通常装扮成一个网络

www.TopSage.com

接口，所以可以用 **ifconfig** 这样的标准配置工具进行操作。有关 **ifconfig** 的更多信息，请参考 12.7 节。

12.13.4 控制 PPP 链路

建立 PPP 连接涉及到事件的确切顺序取决于操作系统和您拨号连接的服务器的类型。连接既可以手工发起，也可以动态发起。

若要手工建立一条 PPP 连接，可以运行一条命令，该命令用调制调解器拨号，登录到远程主机，并且启动远程 PPP 协议引擎。如果这个过程进行得顺利，那么接下来就把串行端口配置为一个网络接口。这个选项一般会将链路保持很长一段时间，这样对专用于 IP 连接的一根电话线最合适。

在动态配置下，守护进程会检查您的串行“网络”接口了解什么时候有流量在排队等候。当有人想发送一个包的时候，守护进程就会自动用调制调解器拨号建立连接、发送包，如果线路又转为空闲状态，那么在适当的时间之后断开连接。如果电话线既传送语音又传送数据流量，或者连接包含长途或连接时间要付费，那么一般都使用动态拨号。

大多数版本的 PPP 都包含实现这两种连接方案的软件。

12.13.5 分配地址

正如必须为以太网上的新主机分配一个 IP 地址那样，也需要为每个 PPP 接口分配一个 IP 地址。将地址分配给这些链路（其中包括根本不分配地址）的方法有很多。在这里我们只讨论最简单的方法。有关分配 IP 地址的更多信息，请参考 12.7 节。

把一条 PPP 链路看作它自己的一个网络。也就是说，只有两台主机的一个网络，这通常叫做一个“点对点”网络。您需要为这个链路分配一个网络号，就像为新以太网段分配一个网络号那样，您可以使用任何在您的网点上有效的规则。您可以在那个网上选出任意两个主机地址，分别分配给该链路的两端。还应该遵循本地的其他规则，如子网划分标准。这样在其他地方看来，每个主机就成为了到达这个点对点网络的一个网关（在现实世界里，您通常不必控制链路的两端，您的 ISP 会提供给您必须在您的一端使用的 IP 地址）。

也可以用 DHCP 在一条 PPP 链路的端点分配地址。有些 ISP 会提供使用 DHCP 的家庭服务，以及花费更大但包含静态地址的商业服务。

12.13.6 路由选择

由于 PPP 要求远程服务器充当一台 IP 路由器，所以需要您像在一个“实际的”网关处（比如，连接两个以太网的计算机）那样考虑 IP 路由。路由选择的目的是指引包通过网关，从而到达最终的目的地。路由选择可以用几种不同的方法配置。有关路由选择的详细信息，请参考第 13 章。

一般的 PPP 客户机应该有一条能把包转发给 PPP 服务器的默认路由。同样，网络上的其他主机应该知道这个服务器，就像叶子（leaf）计算机要知道网关那样。

大多数 PPP 软件包都能自动处理这些路由选择任务。

12.13.7 确保安全性

只要把一台主机加入到一个网络中，就会产生安全性问题。由于通过 PPP 连接的一台主机是网络的真正成员，所以您需要这样处理它们：确保系统没有未设口令或者口令不安全的账号，已安装了相应厂商全部的安全修正补丁等。请参考 12.11 节的内容，了解网络安全的一些特殊问题。Linux 上的 PPP 支持两种验证协议：PAP（Password Authentication Protocol，口令验证协议）和 CHAP（Challenge

www.TopSage.com

Handshake Authentication Protocol，质询握手验证协议）。有关安全性的更多信息，请参考第 20 章。

TopSage.com

12.13.8 使用对话脚本

Linux 串行线路 PPP 实现采用了一个“对话脚本（chat script）”来与调制解调器对话，并登录到远程计算机中启动一台 PPP 服务器。对话脚本由一系列要发送的字符串和期望接收到的字符串所构成，还有一种有限形式的条件语句，能够表达像“期望得到字符串‘Login’，但如果没有得到它，就发送一个回车，再次等候这个字符串”这样的意思。

对话脚本的概念源自于以前 UUCP 的存储-转发系统。在 20 世纪 80 年代，计算机会在半夜叫醒对方，并且通过对话脚本登录进去，然后交换文件。不管这种要求还是否普遍，UUCP 还没有完全销声匿迹：用户 uucp 是 SUSE 上串行设备文件的组属主，您必须是 uucp 组的成员才能将拨出式调制解调器用于 PPP。

大多数 PPP 实现就带有对话脚本的样板，您可以调整它们以适应您自己的环境。您需要编辑这些脚本，设置诸如要拨打的电话号码、在成功之后要运行的命令这样的参数。大多数对话脚本有一个明文的口令，所以要相应地设置好脚本的权限。

12.13.9 Linux 上的 PPP 配置

调制解调器（还有打印机）对于系统管理员来说始终都是一件难以应付的差事。这就不奇怪，为什么在随便拿来的一个调制解调器上配置一条 PPP 连接，用到的软件里有超过 125 种以上的选项——有太多的因素要仔细权衡和配置。

除了 Debian 之外，我们举例的 Linux 发行版本在默认安装中都带了 Paul Mackerras 的 PPP 软件包，它使用一个叫做 **pppd** 的守护进程，而且把它的大多数配置文件都放在**/etc/ppp** 下。运行命令 **pppd --version** 能够看到在您的 Linux 发行版本上安装的 PPP 软件包是什么版本。在 Debian 上可以使用命令 **apt-get install ppp** 来安装这个软件包。

我们介绍的系统都包含了 Roaring Penguin Software 的一个 PPP 版本，它是为在以太网上使用 PPP 而设计的（例如，在一条接到本地 ISP 的 DLS 连接上）。这些系统都包含对 ISDN 连接的 PPP 支持。用于这些介质的配置文件都和那些用于串行链路 PPP 的配置文件一起放在**/etc/ppp** 目录下。它们的文件名除了多加了代表“over Ethernet”的 **oe** 或者代表 ISDN 的 **i** 之外，都比较相似。表 12.16 列出了相应的命令和配置文件的位置。

表 12.16 各个系统上和 PPP 有关的命令和配置文件

系 统	命令或者配置文件	说 明
所有系统	**/usr/sbin/pppd** **/usr/sbin/chat** **/usr/sbin/pppstats** **/usr/sbin/pppdump** **/etc/ppp/options**	PPP 守护进程程序 和调制解调器对话 显示 PPP 链路的统计信息 使 PPP 包成为可读的 ASCII 字符 **pppd** 的配置文件
Debian Ubuntu	**/usr/bin/pon** **/usr/bin/poff** **/usr/bin/plog** **/usr/sbin/pppconfig** **/etc/ppp/peers/provider** **/etc/chatscripts/provider**	启动一条 PPP 连接 关闭一条 PPP 连接 显示 **ppp.log** 末尾的内容 配置 **pppd** **pon** 用于联络 ISP 的选项 **pon** 用于和 ISP 对话的对话脚本
Red Hat（DSL）	**/usr/sbin/pppoe** **/usr/sbin/pppoe-server** **/usr/sbin/pppoe-sniff** **/usr/sbin/adsl-connect** **/usr/sbin/adsl-setup** **/usr/sbin/adsl-start**	PPPoE 的客户端 PPPoE 的服务器 调试提供商特性的嗅探程序 管理链路的脚本 配置 **pppoe** 的脚本 启动 **pppoe** 链路的脚本

www.TopSage.com

续表

系　统	命令或者配置文件	说　明
Red Hat（DSL）	**/usr/sbin/adsl-stop** **/usr/sbin/adsl-status** **/etc/ppp/pppo-e.conf** **/etc/ppp/pppoe-server-options**	关闭 **pppoe** 链路的脚本 显示 **pppoe** 链路的状态 **adsl-***用的配置文件 服务器其他选项的文件
SUSE（DSL）	**/usr/sbin/pppoed** **/etc/pppoed.conf**	PPPoE 客户端 **pppoed** 的配置文件
所有系统（ISDN）	**/usr/sbin/ipppd** **/usr/sbin/ipppstats** **/etc/ppp/ioptions**	PPP over ISDN 守护进程 显示 ISDN PPP 统计信息 **ipppd** 的选项

在我们的配置文件示例中，我们称调制解调器所接的串行端口为**/dev/modem**。有些发行版本实际上有一个**/dev/modem** 文件，该文件是一个链接，链接到系统的某个串行端口上（通常是**/dev/ttyS0** 或者**/dev/ttyS1**），但是这种做法现在已经不用了。您应该用适当的设备文件代替它。参考 27.6 节了解有关串口名称的更多信息。

除了 PPP 软件之外，每种发行版本还提供了几个程序，它们拨号建立一条连接。

既然我们已经讲到了调制解调器的端口和拨号软件，那就让我们谈谈如何设置 **pppd** 来使用它们。全局选项在文件**/etc/ppp/options** 里设置，而特定于连接的选项则保存在目录**/etc/ppp/peers** 和**/etc/chatscripts** 下（在 Debian 和 Ubuntu 上）。Red Hat、Fedora 和 SUSE 倾向于把对话脚本放在目录**/etc/ppp** 里，并用类似 **chat.***remotehost* 这样的名字。Red Hat 也可以在文件**/etc/sysconfig/networkscripts/ifcfg-***ttyname* 里包含某个 PPP 接口上特定于连接的选项。

在默认情况下，**pppd** 首先参考 **options** 文件，然后是用户个人的**~/.ppprc** 启动文件，再看特定于连接的 **options.***ttyname* 文件（如果有这个文件的话），最后看的是它的命令行参数。

Jonathan Corbet 是一位 Linux 老前辈，他提出了一种很方便的窍门，就是定义多个 PPP 接口：一个在家里用，一个在旅行时的饭店里用等。这样的做法能够更容易地切换工作环境。

wvdial 比 **chat** 更聪明，如果没有指定参数，它可以给出合理的默认参数。**wvdial** 从**/etc/wvdial.conf** 得到它的配置信息：调制解调器的细节、登录名、口令、电话号码等。您可以在一个配置文件中给出多个目的地的信息。使用 **wvdialconf** 程序能够推断出您的调制解调器的特性，并为它创建一个初始的 **wvdial.conf** 文件。

下面的配置文件取自几种不同的 PPP 设置。第一个文件**/etc/ppp/options** 设置了 **pppd** 的全局选项。每种发行版本自带的活跃选项如下：



Red Hat 和 **Fedora/etc/ppp/options**:

```
lock
```



SUSE **/etc/ppp/options**:

```
noipdefault
noauth
crtscts
lock
modem
asyncmap 0
nodetach
lcp-echo-interval 30
lcp-echo-failure 4
lcp-max-configure 60
lcp-restart 2
idle 600
noipx
file /etc/ppp/filters
```

www.TopSage.com



Debian 和 Ubuntu：/etc/ppp/options：

```
asyncmap 0
auth
crtscts
lock
hide-password
modem
proxyarp
lcp-echo-interval 30
lcp-echo-failure 4
noipx
```

我们喜欢用下面的选项文件：

```
# Global PPP options
lock                    # Always lock the device you're using
asyncmap 0x00000000     # By default, don't escape anything
crtscts                 # Use hardware flow control
defaultroute            # Add default route thru the ppp interface
mru 552                 # MRU/MTU 512 (data) + 40 (header)
mtu 552
```



接下来是取自 Red Hat 系统上的**/etc/sysconfig/network-scripts/ifcgf-ppp0** 文件的例子。这个文件轮廓是由工具 **linuxconf** 生成的：

```
PERSIST=yes
DEFROUTE=yes
ONBOOT=no
INITSTRING=ATZ
MODEMPORT=/dev/modem
LINESPEED=115 200
ESCAPECHARS=no
DEFABORT=yes
HARDFLOWCTL=yes
DEVICE=ppp0
PPPOPTIONS=
DEBUG=yes
PAPNAME=remote
REMIP=
IPADDR=
BOOTPROTO=none
MTU=
MRU=
DISCONNECTTIMEOUT=
RETRYTIMEOUT=
USERCTL=no
```

下面是和上述 **ifcfg-ppp0** 文件相对应的对话脚本（**chat-ppp0**）的例子（用它简练且有点儿怪异的语法写成）：

```
'ABORT' 'BUSY'
'ABORT' 'ERROR'
'ABORT' 'NO CARRIER'
'ABORT' 'NO DIALTONE'
'ABORT' 'Invalid Login'
'ABORT' 'Login incorrect'
'' 'ATZ'
'OK' 'ATDT phone-number'
```

www.TopSage.com

```
'CONNECT' ''
'TIMEOUT' '120'
'ogin:' 'account'
'ord:' 'password'
'TIMEOUT' '5'
'~--' ''
```

这个对话脚本中有几行包含了一个由一对单引号括起来的空参数，它们看起来和同一字体的一个双引号差不多。

您通常可以改编一个现有的对话脚本，使之适用于您的环境，而无需过多担心它到底是怎样工作的。这里的头几行设定了脚本应该中途退出的几种一般情况。接下来的几行初始化调制解调器、拨号、让线路等待连接，并输入适当的用户名和口令。

有时需要调整对话脚本中的超时设置，以处理较为复杂的拨号情况，比如在有本地电话交换机的酒店或者公司里拨号，或者处理有些电话公司在您给出实际的拨号音之前所用的语音邮件信号。在大多数调制解调器上，电话号码中的逗号都表示在拨号过程中暂停一下。如果您不得不先拨一个特殊的数字，然后等到有第二声拨号音的时候再继续拨号的话，就可能需要用上几个逗号。

在我们的网络上，以 PPP 方式登录的用户名就是在用户名前面加个 P。采用这种约定，很容易就能记起来某台 PPP 机器到底是谁的。

ifup 命令建立起来 **ifcfg-ppp0** 和 **chat.ppp0** 之间的关系，如果 **ifcfg** 文件存在，那么它会在系统启动期间自动执行。如果特定于连接的 options.*ttyname* 文件包含的一个 `connect` 行列出了对应的对话脚本文件名，那么您也可以用它作为参数直接调用 **pppd**。



我们的下一个拨号例子取自 Debian 系统。它使用 **peers** 目录，并把它的对话脚本都放在了目录**/etc/chatscripts** 下，而且使用了 PAP 验证机制，而不是在对话脚本中保存口令。首先是这个连接的选项文件**/etc/ppp/peers/my-isp**：

```
/dev/modem          ### fill in the serial port of your modem
debug
crtscts
name username       ### username at my-isp
remotename my-isp
noauth
noipdefault
defaultroute
connect '/usr/sbin/chat -v -f /etc/chatscripts/my-isp'
```

/etc/chatscripts/my-isp 包含以下几项：

```
'ABORT' 'BUSY'
'ABORT' 'ERROR'
'ABORT' 'NO CARRIER'
'ABORT' 'NO DIALTONE'
'' 'ATZ'
'OK' 'ATDT phonenumber'
'CONNECT' ''
'TIMEOUT' 15
'~--' ''
```

用于连接 ISP 的验证文件**/etc/ppp/pap-secrets** 需要包含这样一行：

```
login-name  my-isp  password
```

www.TopSage.com

这里的 *my-isp* 是前面选项文件中变量 remotename 的值。若要在此情况下启动连接，可以使用命令 **pppd call my-isp**。

下面是在现有的普通 Internet 连接上使用 PPP 的例子，但要配合 **ssh** 在 VPN 上创建一条安全的连接。我们给出了服务器和客户机两端的配置。

服务器的**/etc/ppp/options** 文件：

```
noauth
logfile pppd.log
passive
silent
nodetach
```

每条连接也有一个**/etc/ppp/options.***ttyname* 文件，它包含有给每条连接分配的 IP 地址：

```
local-IPaddress:remote-IPaddress
```

PPP 用户的 shell 设为服务器上的**/usr/sbin/pppd**，这样就会自动启动服务器守护进程。所有的验证密钥都必须事先用 **ssh-agent** 设定，这样就不需要口令了。在客户机一端，配置是在**/etc/ppp/peers** 目录下的一个文件里完成的，针对服务器来起这个文件名字——我们称这个配置文件为“my-work”。客户机的**/etc/ppp/peers/my-work** 文件包含的内容是：

```
noauth
debug
logfile pppd.log
passive
silent
pty "ssh -t user@remotehost"
```

若要从家里通过一条安全 PPP 连接登录进来，用户只要键入 **pppd call my-work** 就行了。

最后，我们给出一个例子，它使用 **wvdial** 命令和它的简易配置来避免所有那些看似必要的对话脚本戏法：

```
[Dialer Defaults]
Phone = phonenumber
Username = login-name
Password = password
Modem = /dev/ttyS1

[Dialer creditcard]
Phone = long-distance-access-code,,,phone-number,,cc-number
```

如果不带参数调用 **wvdial**，它就用**/etc/wvdial.conf** 文件中的 dialer defaults（拨号程序默认）段的配置，或者您的**~/.wvdialrc** 文件中的配置来发起拨号并启动 PPP。如果带参数调用它（例如，**wvdial creditcard**），它就用配置文件中适当的配置段覆盖在默认段中指定的任何参数。

若要让一条 PPP 连接关闭，最好用 **ifdown** 命令，而不仅仅是杀死 **pppd** 守护进程就行了。如果您直接杀死了 **pppd**，Linux 会注意到这一点并重新启动它。

```
$ sudo ifdown ppp0
```

如果您用的是便携机，有时候会使用以太网而不是 PPP，那么在 **pppd** 启动之前，可能会有一条通过以太网接口的默认路由。遗憾的是，**pppd** 太有礼貌了，它不会删除那条路由并安装一条它自己的默认路由，这可能并不是您确实想要的结果。要修正这个问题，可以在您启动 **pppd** 之前删除现有的默认路由。最简单的做法是写一个您自己的 **ppp.up** 脚本，来自动删除默认路由。

下面是在 PPP 连接已经建立起来之后，PPP 接口的配置和路由表的内容：

www.TopSage.com

大家网 TopSage.com

```
$ ifconfig ppp0
ppp0  Link encap:Point-to-Point Protocol
      inet addr:10.0.0.56  P-t-P:10.0.0.55  Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:125 errors:0 dropped:0 overruns:0 frame:0
      TX packets:214 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:3
      RX bytes:11446 (11.1 Kb)  TX bytes:105586 (103.1 Kb)

$ netstat -nr
Kernel IP routing table
Destination  Gateway    Genmask         Flags  MSS  Window  irtt  Iface
10.0.0.55    0.0.0.0    255.255.255.255 UH      40  0          0  ppp0
0.0.0.0      10.0.0.55  0.0.0.0         UG      40  0          0  ppp0
```

您可以用 **pppstats** 命令获得有关 PPP 连接和它已经传送的包的统计信息：

```
$ pppstats
   IN  PACK VJCOMP VJUNC VJERR |     OUT PACK VJCOMP VJUNC NON-VJ
11862  133      8     96      0 | 110446  226     27     89     110
```

VJCOMP 列计算了使用 Van Jacobson 的 TCP 包头压缩算法的包数，VJUNC 列计算了没有使用压缩的包数。参考 RFC1144 了解详细情况。

调试一条 PPP 连接真是一件很痛苦的事情，因为要牵扯到那么多的因素。在 Red Hat 和 Debian 系统上，**pppd** 通过 syslog 向 daemon 设备（facility）提交日志项，而在 SUSE 系统上，则向 local2 设备提交日志项。您可以通过在 **pppd** 的命令行上使用 **debug** 标志，或者通过在选项文件中要求有更多的日志来提高日志级别。**pppd** 还提供了故障时的退出代码（exit code），所以，如果您试着执行 **pppd**，而它不肯运行的话，就可以运行 **echo $status** 命令（在您做其他任何事情之前）以重新得到退出代码，然后在 **pppd** 的手册页里查一下这个值。



SUSE 会包含每个子系统的配置文件样本，它们的内容大部分都是注释，说明了格式和可用选项的含义。在 SUSE 的目录**/etc/ppp** 下的这些文件无一例外，它们都有很好的文档，而且有许多参数的合理的建议值。



Debian 也有文档详尽的 PPP 配置文件样板。它有一个专门用于对话脚本的子目录**/etc/chatscripts**。若要在一个接口上启动 PPP，您可以在文件**/etc/network/interfaces** 中用 PPP 方法加上这个接口，并带 provider 选项把您的提供商的名字（在我们的例子里是 my-isp）和**/etc/peers** 目录下的一个文件名（**/etc/peers/my-isp**）联系起来。例如：

```
iface eth0 inet ppp
      provider my-isp
```

在这种情况下，要用 Debian 特有的命令 **pon** 和 **poff** 来管理连接。

12.14 Linux 连网配置的特殊之处

Linux 同大多数内核不一样，它会注意 IP 包中的 TOS（type-of-service，服务类型）域，并且会向被标记为交互式（低延迟）的包提供更快的服务。遗憾的是，微软方面犯的错误却让您不得不关闭这项完全合理的功能。

www.TopSage.com

源自 Windows 老系统的所有包，不管它们的用途是什么，都被标记成了交互式的包。另一方面，UNIX 系统通常不把任何包标记成交互式的。如果您的 Linux 网关是向一个既有 UNIX 又有 Windows 系统的混合网络提供服务，那么 Windows 的包就一定会得到优先处理。这对 UNIX 的性能所造成的影响相当明显。

在您编译 Linux 内核的时候，可以关闭基于 TOS 对包进行分类的功能。只要不选"IP: use TOS value as routing key"这项就行了。

在启用了 IP 伪装（也就是 NAT）的时候，它就告诉内核，在把包转发出去之前，先把包的分片组装成一个完整的包，即使内核必须立即对包重新分片，才能把它发送出去也是如此。重新组装颇费 CPU 周期，但是在现代计算机上，CPU 的速度足够快，所以这确实不是个大问题了。

Linux 允许您改变某些类型的网络接口的 MAC 级地址：

```
redhat$ ifconfig eth1
eth1   Link encap:Ethernet  HWaddr 00:02:B3:19:C8:87
       BROADCAST MULTICAST  MTU:1500  Metric:1
       RX packets:0 errors:0 dropped:0 overruns:0 frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       Interrupt:7 Base address:0xee80

redhat$ sudo ifconfig eth1 hw ether 00:02:B3:19:C8:21
redhat$ ifconfig eth1
eth1   Link encap:Ethernet  HWaddr 00:02:B3:19:C8:21
       BROADCAST MULTICAST  MTU:1500  Metric:1
       RX packets:0 errors:0 dropped:0 overruns:0 frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       Interrupt:7 Base address:0xee80
```

这是一项危险的功能，它会有破坏作用。它能带来方便，但只应该在万不得已的时候才用。

12.15 推荐读物

STEVENS, W. RICHARD. *TCP/IP Illustrated, Volume One: The Protocols.* Reading, MA: Addison-Wesley. 1994.

WRIGHT, GARY R., AND W. RICHARD STEVENS. *TCP/IP Illustrated, Volume Two: The Implementation.* Reading, MA: Addison-Wesley. 1995.

这两本书是介绍 TCP/IP 协议栈全面彻底的指导书籍。虽然稍有一点儿过时，但还是很可靠的。

STEVENS, W. RICHARD. *UNIX Network Programming.* Upper Saddle River, NJ: Pren tice Hall. 1990.

STEVENS, W. RICHARD. *UNIX Network Programming, Volume 1: The Sockets Networking API (3rd Edition).* Upper Saddle River, NJ: Prentice Hall. 2003.

STEVENS, W. RICHARD. *UNIX Network Programming, Volume 2: Interprocess Communications (2nd Edition).* Upper Saddle River, NJ: Prentice Hall. 1999.

这几本书都是学生上网络编程课的经典教材。如果您只需要用到 Berkeley 套接口，那么第一版就已经是很好的参考书了。如果您还需要用到 STREAMS 接口，那么就看第三版。这一版还包括有关 IPv6 的内容。3 本书都是以典型的 Rich Stevens 风格编写的。

TANENBAUM, ANDREW. *Computer Networks, (4th Edition).* Upper Saddle River, NJ: Prentice Hall. 2003.

这本书是第一本讲网络技术的教科书，也是一本经典。它非常详细地介绍了协议栈的物理层和链

www.TopSage.com

路层的知识。最新一版的内容涵盖了无线网络、千兆以太网、P2P 网络和 VoIP 等技术。

SALUS, PETER H. *Casting the Net, From ARPANET to INTERNET and Beyond.* Reading, MA: Addison-Wesley. 1995.

这本书讲述了从 ARPANET 成长为 Internet 的有趣历史，作者是一位历史学家，他长期和 UNIX 人士保持着接触，仿佛自己也是他们中的一员！

COMER, DOUGLAS. *Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architectures, (5th Edition).* Upper Saddle River, NJ: Prentice Hall. 2006.

Doug Comer 的 *Internetworking with TCP/IP* 系列图书一直以来就是 TCP/IP 协议的标准参考书。这些书是按本科生的教科书来编写的，是背景资料的好来源。

HEDRICK, CHARLES. "Introduction to the Internet Protocols." Rutgers University. 1987

这份文档缓缓地介绍了 TCP/IP 协议。它似乎没有一个稳定的家，但是在网上到处都能找到，搜索一下吧。

HUNT, CRAIG. *TCP/IP Network Administration, (3rd Edition).* Sebastopol, CA: O'Reilly & Associates. 2002.

和 nutshell 系列的其他书籍一样，这本书也是面向 UNIX 系统的管理员而写的。这本书有一半的内容和 TCP/IP 有关，而剩下的一半则介绍更高层的 UNIX 工具，比如电子邮件和远程登录等。

在 www.isoc.org/internet/history 汇集了有关 Internet 历史及其各种技术的文档。

12.16 习题

E12.1 听从（也就是服从）ICMP 重定向包的指挥，如何会让一个未经授权的用户威协网络的安全？

E12.2 网络链路的 MTU 是什么？如果给定链路的 MTU 设得太高，会发生什么问题？太低呢？

★E12.3 说明划分子网的概念，解释它有用的原因。什么是网络掩码？网络掩码同 IP 地址的网络部分和主机部分之间的划分有什么关系？

★E12.4 网络 134.122.0.0/16 已经细分成了几个/19 网络。

a）有多少个/19 网络？列出它们。它们的网络掩码是什么？

b）每个子网上有多少台主机？

c）IP 地址 134.122.67.124 属于哪一个子网？

d）每个网络的广播地址是什么？

★E12.5 网络 128.138.2.0/24 上的主机 128.138.2.4 向网络 128.138.129.0/24 上的主机 128.138.129.12 发送一个包。假定：

- 主机 128.138.2.4 有一条通过 128.138.2.1 的默认路由；
- 主机 128.138.2.4 刚启动，还没有发送或者接收到任何包；
- 网络上其他所有机器都已经运行很长一段时间了；
- 路由器 128.138.2.1 直接有一条到 128.138.129.1 的路由，后者是子网 128.138.129.0/24 的网关。

a）列出发出这个包所需的所有步骤。给出所有传输包的源目的以太网和 IP 地址。

b）如果网络是 128.138.0.0/16，您的答案会变吗？怎么变？

c）如果网络 128.138.2.0 是一个/26 网络，而不是/24 网络，您的答案会变吗？怎么变？

★★E12.6 在安装了一个新的 Linux 系统之后，您会如何解决本章里提到的若干安全问题？查

www.TopSage.com

查看，在您实验室的 Linux 系统上是否有什么要解决的安全问题（可能需要 root 权限）。

★★E12.7 在您实验室环境里的网络加入一台新机器需要哪些步骤？回答的时候，要用适合于您的网络和本地情况的参数。假定新机器已经在运行 Linux 了。

★★E12.8 给出设置一台可以分配 128.138.192.[1-55]范围内 IP 地址的 DHCP 服务器所需的配置文件。用到的租期为 2 小时，要确保以太网地址为 00:10:5A:C7:4B:89 的主机始终会分配到 IP 地址 128.138.192.55。

www.TopSage.com

第 13 章　路由选择



保持对网络流量的跟踪并不是一件轻松的工作。第 12 章简要介绍了 IP 包的转发机制。我们将在这一章更详细地考察转发过程，并研究几种能让路由器自动发现有效路由的网络协议。路由协议不仅减轻了维护路由信息的日常管理负担，而且，如果路由器或者网络出现故障，它们还可以迅速重新指引网络流量。

虽然我们把实际转发 IP 包的过程和推进这个过程的路由表管理工作都称为“路由选择（routing）”，但把二者区分开是很重要的。包的转发很简单，然而计算包的路由却需要相当的技巧，因此，第二个意思在实际中更常用。本章只介绍单播路由，多播路由涉及一系列截然不同的问题，这超出了本书的范围。

对于绝大多数情形而言，本书第 12 章介绍的内容，就是您要掌握的全部路由知识。如果合适的网络基础设施已经到位，那么您只需要设置一条静态路由（按照 12.5 节的介绍），您已经获得足够的信息，能够访问到 Internet 上的任何地方。如果您必须在一个复杂的网络拓扑结构中正常使用网络，或者如果您正在把一个 Linux 系统作为网络基础设施的一部分来用，那么本章里关于动态路由协议和工具的内容就会大有用武之地。

一般人习惯性地认为 IP 路由出奇的困难，只有生活在加州校园里劳伦斯伯克利实验室地下暖气隧道中的少数几位长发奇才能理解。实际上，只要理解 IP 路由就是“下一跳”路由这一基本前提，那么 IP 路由就不会如此困难。在任何给定的地点，您只需要确定包向目标传送道路上的下一台主机或路由器即可。这种方法与许多老式的协议不同，老式协议在一个包离开它的源主机之前，就已经确定了它将要经过的准确路由，这叫做源路由[1]。

[1] 源路由 IP 包是可能的，但几乎从未使用过。因为考虑到了安全因素，所以这种特性并没有得到广泛支持。

www.TopSage.com

TopSage.com

13.1 近观包转发

在开始介绍路由表的管理之前，首先应当详细了解怎样使用这个表。我们看一下图 13.1 中展示的网络。

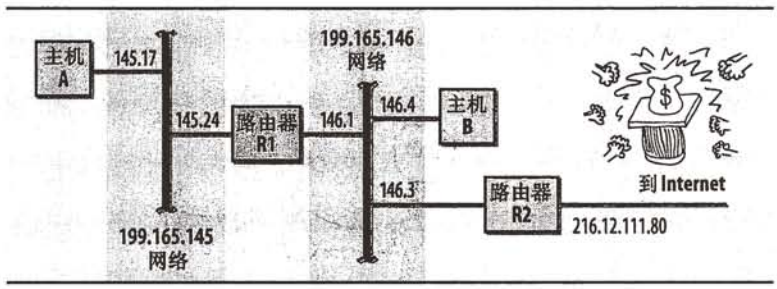


图 13.1 网络示例图

路由器 R1 连接两个网络，而路由器 R2 将其中一个连接到外界（眼下我们假定 R1 和 R2 是 Linux 计算机，而不是专用路由器）。让我们看一些路由表和一些具体的包转发情况。首先，主机 A 的路由表为：

```
A$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window  irtt Iface
199.165.145.0   0.0.0.0         255.255.255.0   U       0 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U       0 0          0 lo
0.0.0.0         199.165.145.24  0.0.0.0         UG      0 0          0 eth0
```

在 4 台机器中，主机 A 的路由配置是最简单的。前两条路由用标准的路由格式描述了机器自身的网络接口。因为有了这两项，所以不需要把到直连网络上的转发作为特例来处理。eth0 是主机 A 的以太网接口，而 lo 是环回接口，它是软件模拟的一个虚拟网络接口。通常，当配置网络接口时由 **ifconfig** 自动添加这两项。关于 **ifconfig** 的更多信息，请参见 12.7.2 节。

主机 A 上的默认路由将目标地址既不是环回地址也不是 199.165.145 网络的所有包转发到路由器 R1（R1 在这个网络上的地址是 199.165.145.24）。G 标志表明这条路由通向网关，而不是通向主机 A 的某个本地接口。到网关必须只有一跳。

假定 A 上的一个进程要向 B 发送一个包，B 的地址是 199.165.146.4。IP 实现要找出一条到目标网络 199.165.146 的路由，但没有一条路由能匹配。它便调用默认路由，并把包转发到 R1。图 13.2 显示了实际送出到以太网上的帧（以太网帧头的地址是 A 的 MAC 地址以及 R1 在 145 网络上接口的 MAC 地址）。有关以太网和 IP 寻址的更多信息，请参见 12.3.2 节。

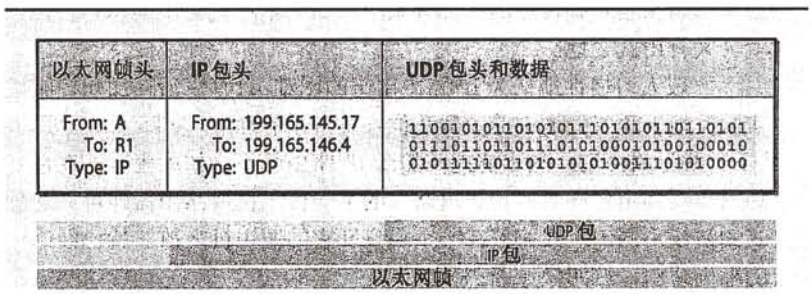


图 13.2 以太网上的包

www.TopSage.com

以太网目的地的硬件地址是路由器 R1 的硬件地址，但隐藏在以太帧中的 IP 包根本就没有提到过 R1。当 R1 检查它接收到的包时，它从 IP 目的地址中发现此地址并不是包的最终目的地。于是，它就用自己的路由表来把包转发到主机 B，而且并不重写 IP 包头，这样这个 IP 的包头仍然显示包是来自 A 的。

以下是 R1 的路由表：

```
R1$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window  irtt Iface
127.0.0.0       0.0.0.0         255.0.0.0       U       0 0          0 lo
199.165.145.0   0.0.0.0         255.255.255.0   U       0 0          0 eth0
199.165.146.0   0.0.0.0         255.255.255.0   U       0 0          0 eth1
0.0.0.0         199.165.146.3   0.0.0.0         UG      0 0          0 eth1
```

除了表中有两个物理网络接口外，此表与主机 A 的路由表很相似。在本例中，默认路由指向 R2，因为 R2 是到达 Internet 要经过的网关。发往任何一个 199.165 网络的包都能够直接送达。

像主机 A 一样，主机 B 也只有一个真实的网络接口。然而，因为主机 B 直接连到两个不同的路由器，所以为了正常运行，它还得多要一条路由。到 199.165.145 网的流量必须经过 R1，而其他流量经过 R2 送到 Internet 上。

```
B$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window  irtt Iface
127.0.0.0       0.0.0.0         255.0.0.0       U       0 0          0 lo
199.165.145.0   199.165.146.1   255.255.255.0   U       0 0          0 eth0
199.165.146.0   0.0.0.0         255.255.255.0   U       0 0          0 eth0
0.0.0.0         199.165.146.3   0.0.0.0         UG      0 0          0 eth0
```

开始只知道一个网关也有可能配置主机 B，这样要借助于 ICMP 重定向的帮助，以消除多余的跳数。参考 12.5.2 节了解 ICMP 重定向的说明。例如，对于主机 B，一种可能的初始配置是：

```
B$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window  irtt Iface
127.0.0.0       0.0.0.0         255.0.0.0       U       0 0          0 lo
199.165.146.0   0.0.0.0         255.255.255.0   U       0 0          0 eth0
0.0.0.0         199.165.146.3   0.0.0.0         UG      0 0          0 eth0
```

如果 B 接着要发送一个包给主机 A（199.165.145.17），没有匹配的路由，包被转发到 R2 进行传递。R2（它作为路由器，假定掌握着关于网络的全部信息）将把包发送到 R1。因为 R1 和 B 在同一个网络上，所以 R2 也向 B 发送一个 ICMP 重定向的通知，B 就在它的路由表中为 A 加入一条主机路由：

```
199.165.145.17 199.165.146.1 255.255.255.255 UGHD 0 0      0 eth0
```

这条路由将通过 R1 直接为 A 发送以后的所有流量。然而它并不会影响到 A 网络上其他主机的路由机制，它们必须借助于从 R2 的单独重定向来发送。

考虑到 ICMP 重定向具有的动态特性，因此，有些网点选择此方法作为它们的主要路由“协议”。遗憾的是，一旦内核从一个重定向那里学到了一条路由，而如果重定向信息改变了，就必须手工删除原来的路由，或者必须重新启动主机。因为这个问题和重定向的其他几个缺陷（增加网络负载、增加 R2 上的负载、路由表混乱、依赖额外的服务器），建议不要在配置上这样使用重定向。在配置合理的网络中，决不应该在路由表中出现重定向。

www.TopSage.com

13.2 路由守护进程和路由协议

在图 13.1 所示的简单网络中，手工配置路由是完全合理的。然而，在某些情况下，网络变得相当复杂，以至于不能用这种方法进行管理（可能是因为网络增长的速度很快）。我们并不需要明确告诉网络上的每个计算机怎样到达其他计算机和网络，而是让计算机们能够碰碰头来解决这个问题那就好了。这就是路由协议以及实现路由协议的守护进程的工作。

能够对变化的网络情况做出反应，并适应网络变化的情况，是路由协议相对于静态路由系统的一个主要优势。一条链路出现故障时，如果有任何替代路由存在的话，那么路由守护进程就能够迅速发现并向该链路所服务的网络公告替代路由。

路由守护进程从 3 个来源收集信息：配置文件、现有的路由表和其他系统上的路由守护进程。它将这些信息合并起来计算出最佳的一组路由，然后把这些新路由反馈到系统路由表中（通过一种路由协议还可能反馈到其他系统上）。因为网络情况不断变化，所以路由守护进程相互之间必须定期检查，以确保它们的路由信息仍然是当前的并未过时。

计算路由的正确方法取决于路由协议。常用的有两种类型的协议：距离向量协议和链路状态协议。

13.2.1 距离向量协议

距离向量协议（也称之为“传话式”协议）以这样的基本思想为基础，即“如果路由器 X 距离网络 Y 有 5 跳，并且我与路由器 X 相邻，那么我距离网络 Y 必有 6 跳”。您宣布您认为自己距离您所知道的网络有多远。如果您的邻居没有更好的途径到达每个网络，那么他们就会把您作为最好的网关。如果您的邻居已经知道一条更短的路由，那么他们就会忽略您的公告[2]。随着时间的推移，每个人的路由表都应该收敛到一种稳定的状态。

距离向量协议确实是一个非常优雅的想法。如果它能像其声称的那样工作，那么路由工作就会相当简单。遗憾的是，这类算法并不能够很好地处理拓扑结构中的变化。在一些情况下，无限循环（例如，路由器 X 接收来自路由器 Y 的信息，并把信息发送给路由器 Z，而路由器 Z 又把信息发送回路由器 Y）会阻止路由收敛。实际中的距离向量协议必须通过一些方法来避免这样的问题，这些方法有，引进复杂的试探机制，或者是强加诸如任何一个超过 15 跳的网络都是不可达的这样的 RIP（Routing Information Protocol，路由选择信息协议）概念来进行限制。

即使在没有路径逻辑的情况下，所有路由器也能够用多个更新周期来达到稳定的状态。因此，为了确保在更长的一段时间内路由不会堵塞，必须缩短更新周期的时间，由于这个原因，距离向量协议一般广播量很大。例如，RIP 要求路由器每隔 30 秒广播所有路由信息。IGRP 和 EIGRP 每隔 90 秒发送更新信息。

另一方面，BGP（Border Gateway Protocol，边界网关协议）要传送整个路由表一次，以后在每次路由发生变化时，再传送变化。这种优化实质上减少了潜在的“闲聊”（并且大部分是不必要的）流量。

表 13.1 列出了现在常用的距离向量协议。

[2] 实际情况并不是这样简单，因为有一些处理拓扑结构变化的规定，这些规定可能会使现存的路由变长。EIGRP 这样的一些 DV（距离向量）协议保存了多条可能路由的信息，于是它们总是有退一步的方案。确切的细节并不重要。

www.TopSage.com

表 13.1 常用的距离向量路由协议

协 议	全 名	应 用
RIP	Routing Information Protocol（路由选择信息协议）	内部 LAN
IGRP	Interior Gateway Routing Protocol（内部网关路由选择协议）	小型 WAN
EIGRP	Enhanced Interior Gateway Routing Protocol（增强型内部网关路由选择协议）	WAN、企业级 LAN
BGP	Border Gateway Protocol（边界网关协议）	Internet 主干路由

13.2.2 链路状态协议

链路状态协议以一种相对不作处理的形式发布信息。在路由器之间进行交换的记录形式是“路由器 X 与路由器 Y 相邻，并且二者之间的链接已经建立”。一整套这种形式的记录形成了网络的连通图，根据这份连通图，每个路由器能够计算出自己的路由表。与距离向量协议相比，链路状态协议最基本的优势是：发生事故后，它能够迅速找到一个可操作的路由方案。但相比之下链路状态协议的缺点是：在每个网点维护网络的完整的“地图”需要消耗内存和 CPU，而在距离向量路由系统中则不需要。

因为在链路状态协议下，路由器之间的通信不是实际的路由计算算法的组成部分，所以能够在不发生传输环路的情况下实现通信。只需在网络带宽和 CPU 时间上的较低开销，就能高效地在网络上公告拓扑数据库的更新。

链路状态协议比距离向量协议更复杂，这可以部分地用下面的事实来解释：在链路状态协议下，更容易实现一些高级特性，例如服务类型路由和到同一目的地的多重路由。目前，普通的 Linux 系统都不支持这些特性，要想充分发挥它们的好处，则必须使用专门的路由器。

表 13.2 列出了常见的链路状态协议。

表 13.2 常用的链路状态协议

协 议	全 名	应 用
OSPF	Open Shortest Path First（开放最短路径优先）	内部 LAN、小型 WAN
IS-IS	Intermediate System to Intermediate System（中间系统到中间系统，IS-IS）	实验室做实验，精神病院

13.2.3 代价度量

路由协议为了确定哪条路是最短的，必须定义“最短”的意思[3]。这条路由是否有最少的跳数？路径的延迟时间是否最小？是否有最大的最小中间带宽？资金成本是否最低？

对于路由选择而言，链路质量用一个叫做代价度量的数字来表示。把一条路径上的各段链路的代价加到一起，就能计算出这条路径的代价。在最简单的系统中，如果每段链路的代价是 1，就相当于把跳数作为一个路径度量。但我们也能够把上述提到的任何一种因素转变为一种数值上的代价度量。

很长时间以来，网络专家们一直努力想给代价的度量下一个灵活的定义，而且一些现代的协议也允许对不同类型的网络流量使用不同的度量。然而，在 99%的情况下，所有这些费劲儿的任务都可以不做，也没问题。对多数系统而言，默认度量就不错。

您可能会遇到这样的情况，因为某些政治上的原因，到一个目标的实际最短路由并不是很好的默认路由。为了解决这种情况，可以人为地提高关键链路的代价，使这样的关键链路不那么吸引人。现在我们暂且不管路由配置的其他方面。

[3] 幸好不必定义“is”的意思。

www.TopSage.com

13.2.4 内部和外部协议

“自治系统（AS，Autonomous System）”是指：处于一个单独实体的管理和行政控制之下的一组网络。这个定义是非常模糊的；实际的自治系统可能大到是全球性的企业网络，也可能小到是一座建筑物或学院中的一个系的网络。这都取决于管理员想怎样来管理路由。一般倾向于使自治系统尽可能地大。这种习惯能让管理更简单，并且使路由尽可能地高效。

自治系统的内部路由与自治系统之间的路由有些不同。自治系统（AS）之间的路由协议（“外部”协议）必须经常处理去往许多网络的路由，并且必须妥善地处理这样的实际情况，即临近的路由器是处于他人的控制之下。外部协议不能暴露自治系统内部的拓扑结构，因此，可以从某种意义上把外部协议看作是路由体系的第二层，它们处理网络的集合，而不是个别的主机或电缆。

实际上，对于中小规模的网点来说，除非它们与多个 ISP 相连接，基本上都不需要运行外部协议。在有多家 ISP 的情况下，网络可以简单地划分为本地域和 Internet 域，路由器必须确定对任何一个特定的地址，通向 Internet 的哪条路由是最好的（不过，这并不是说每台路由器都必须知道这一信息。许多主机仍然可能对此信息一点都不了解，而是通过它的内部网关来发送自己的默认包，内部网关知道的信息更多）。

虽然外部协议与内部协议的差异并不是特别大，但本章主要讲述内部协议以及支持内部协议的守护进程。如果您的网点还必须使用外部协议，那么建议您查看 13.8 节，以得到一些有建议性的参考资料。

13.3 内部路由协议巡礼

常用的内部路由协议有几种。本节介绍几个主要的内部路由协议，并总结它们主要的优缺点。

13.3.1 RIP：路由信息协议

在 RFC1058 中定义的 RIP 是一种老的 Xerox 协议，但它已经调整为能够适用于 IP 网络。它是由 **routed** 使用的协议。RIP 是一种简单的距离向量协议，它使用跳数作为代价度量。因为在设计 RIP 的时代，单台计算机的价格高达数十万美元，同时相对现在而言，那时的网络又很小，所以 RIP 认为，任何在 15 或更多跳数以外的主机都是不可达的。因此，如果某个本地较大网络中任何一条路径上的路由器超过 15 个，那么就不能使用 RIP 协议。

虽然 RIP 过度使用广播，使得它很占资源，但如果网络经常变化，或者不了解远程网络的拓扑结构时，RIP 却能很好地起作用。然而，在一条链接出现故障后，它可能要慢慢地才能稳定下来。

许多网点使用 **routed** 的**-q**（“沉默”）模式，在这种模式下，**routed** 管理路由表并监听网络上的路由更新，但它却不广播关于自己的任何信息。在这些网点上，通常使用像 OPSF（参见下一节的内容）这样更有效的协议来完成实际的路由计算。计算得出的路由被转为 RIP 更新，以供不是路由器的机器使用。**routed** 很轻便（在**-q** 模式下）并且受到广泛的支持，所以大多数主机不必做特殊配置，就能享受到动态路由选择的好处。

RIP 在非 Linux 平台上得以广泛实现。从打印机到 SNMP 可管理的网络部件，许多这样的常见设备都能够监听 RIP 的公告来了解可能有的网关。除此之外，几乎所有版本的 UNIX 和 Linux 都有 **routed**，因此，RIP 是一种事实上最通用的路由协议。RIP 经常用于 LAN 的路由选择，而用于广域连接的话，它是一种功能较强的协议。

www.TopSage.com

大家网 TopSage.com

13.3.2 RIP-2：路由选择信息协议第 2 版

RIP-2 对 RIP 做了适度修改，增加了对原协议中没有的几个特性的支持。最重要的变化是：RIP-2 把下一跳的地址和网络掩码一起发布，因此它对划分子网的网络和 CIDR 的支持比 RIP 更好。在提高 RIP 安全性上，RIP-2 也有了一个模糊的表示，但是具体认证系统的定义则仍然留待以后再开发。关于无类地址（即 CIDR）的更多信息，请参考 12.4.4 节。

RIP-2 提供了几个看起来是面向多协议环境运行的特性。“下一跳（next hop）”更新能够让广播方公告那些并非用于真正网关的路由，而且“路由标志（route tags）”允许通过 RIP 来公告外部发现的路由。

RIP-2 能够在兼容模式下运行，这个模式保留了 RIP-2 的大多数新特性，但又没有完全抛弃普通的 RIP 接收方。RIP-2 在许多方面都与 RIP 相同，但如果正在使用的系统支持 RIP-2，那么应该优先选择 RIP-2。但是，Linux 发行版本一般不直接支持它。

13.3.3 OSPF：开放最短路径优先

OSPF 是在 RFC2328 中定义的。它是一个链路状态协议。“最短路径优先”是指用于计算路由的数学算法；“开放”在此有“非专有”的意思。

OSPF 是第一个得到广泛使用的链路状态协议，而且它现在仍然是最流行的链路状态协议。它之所以得到广泛应用，很大原因在于 **gated** 支持它。**gated** 是一个流行的多协议路由守护进程，在后面我们将会详细介绍这个程序。但遗憾的是，这种协议本身很复杂，因此只有在网络规模很大的地方（此处，路由协议的行为确实会有差异）才值得使用这个协议。

OSPF 协议规范不要求任何特殊的代价度量。在默认情况下，Cisco 的实现使用跳数作为代价度量。也可以配置成使用网络带宽作为代价度量。

OSPF 是一种具有业界实力的协议，在大型而复杂的拓扑结构中运行得非常好。它比 RIP 有这样一些优点：能够管理到达一个目标的几条路由，能够把网络划分成只共享高级路由信息的多个部分(“区域[area]”)。

13.3.4 IGRP 和 EIGRP：内部网关路由协议

IGRP 和它具有更强功能的后续协议 EIGRP 都是只能在 Cisco 路由器上运行的专有路由协议。在像 OSPF 这样健壮的标准存在之前，创建 IGRP 的目的是为了解决 RIP 的一些缺点。虽然在许多地方还在使用 IGRP，但是现在它已经被 EIGRP 取代。虽然在底层协议的设计上，EIGRP 与 IGRP 实际上很不一样，但二者的配置却很相似。IGRP 只按传统的 IP 地址类的边界来处理路由通告，但 EIGRP 能够理解任意的 CIDR 网络掩码。

虽然 IGRP 和 EIGRP 都是距离向量协议，但是它们的设计避免了在其他 DV（距离向量）系统中出现的环路和收敛问题。人们普遍把 EIGRP 看作是距离向量协议的典范。对于大多数目的来说，EIGRP 和 OSPF 在功能上是等效的。

笔者认为，最好坚持使用一种像 OSPF 这样既定的、非专有的、有多种实现的路由协议。更多的人正在使用并在上面工作的是 OSPF 而不是 EIGRP，OSPF 有几种实现。

13.3.5 IS-IS：ISO 的“标准”

IS-IS（Intra-Domain Intermediate System to Intermediate System Routing Protocol，域内中间系统到

www.TopSage.com

中间系统路由协议）是国际标准化组织对 OSPF 的回应。设计它的最初目的是：为 OSI 网络协议管理“路由（routeing）”，后来又被扩展为处理 IP 路由（routing）。（译者注：美国英语和英国英语在路由器“router”一词的发音上不同，美国英语发/ow/的音，而英国英语发/uu/的音。在表达路由器功能的单词上，两者也在拼写和读音上出现了不同。美国英语用“routing”的拼法，发音也是/ow/，而英国英语用“routeing”的拼法，发音保持为/uu/。由于 ISO 和 CCITT 的文件中大多使用英国英语，而 IETF 的 RFC 则基本采用美国英语，所以就有了上面的差异。针对现在网络界，IETF 的标准和影响不断扩大，而 ISO 和 CCITT 则日益势微的现状，身为美国人的作者显然在这里流露出了一丝得意的口气。）

IS-IS 和 OSPF 都是在 20 世纪 90 年代前期开发的，当时的 ISO 协议在政治上相当时髦。IETF 早先对 IS-IS 的关注帮助 IS-IS 沾上了有 IP 合法外表的光，但看起来它远没有 OSPF 流行。现如今，除了厂商的认证测试环境之外，几乎见不到用 IS-IS 的地方了。这个协议本身也有许多 ISO 的毛病，一般应该避免使用它。

13.3.6 MOSPF、DVMRP 和 PIM：多播路由协议

设计 MOSPF（多播 OSPF）、DVMRP（Distance Vector Multicast Routing Protocol，距离向量多播路由协议）和 PIM（Protocol Independent Multicast，协议无关多播）协议是为了支持 IP 多播，多播是还没有广泛使用的一项技术。可以在 www.mbone.com 站点找到关于这些协议的更多信息的链接。

13.3.7 路由器发现协议

路由器发现协议（Router Discovery Protocol）使用 ICMP 消息（这些消息发送到 IP 多播地址 224.0.0.1）来通知和了解网络上的其他路由器。遗憾的是，不是所有路由器都发这类通告，也不是所有主机都监听通告。希望有一天这个协议会变得更流行。

13.4 routed：使用 RIP 的路由守护进程

您可能没有钱，您也可能长得不好看，但您一定会有 **routed**。很长时间以来，**routed** 一直是标准的 UNIX 路由守护进程，现在大多数版本的 UNIX 和 Linux 仍然提供了 **routed**。

Linux 的 **routed** 只用 RIP。如果您打算使用 RIP-2，那么从 sourceforge.net/projects/nx-routed 下载的 Nexus Routing Daemon 是个容易配置的选择。只有您的子网没有按字节边界掩码的情况下，才会非要用 RIP-2 不可。

routed 既能够在服务器模式（**-s**）下运行，也能够在沉默模式（**-q**）下运行。两种模式都监听广播，但只有服务器模式发布它自己的信息。通常，只有那些具有多个接口的机器才应该是服务器。如果既没有指明**-s**，也没有指明**-q**，那么 **routed** 应该在只有一个接口时以沉默模式运行，而在具有多个接口时以服务器模式运行。

routed 把它发现的路由添加到内核的路由表中。至少每隔 4 分钟就必须复审路由，否则路由将被删除。不过，**routed** 知道它已经添加了哪些路由，而且不会删除用 **route** 命令安装的静态路由。有关 **route** 的更多信息，请参考 12.7.4 节。

routed -t 用来调试路由。这个选项让 **routed** 在前台运行，并把它发送或接收的包都输出到屏幕。

一般而言，**routed** 能够动态地发现路由信息，它不要求配置。然而，如果网点包含了到 Internet 或其他自治系统的网关，那么必须采取额外的步骤来使 **routed** 可以同这些链路配合使用。

如果只有一个出口网关，那么，使用**-g** 标志来运行 **routed**，可以将这个网关公告为一条全局默认

www.TopSage.com

路由。这与在一台机器上设置默认路由类似，不同之处在于，这条路由会传遍您的网络。

routed 还支持一个配置文件**/etc/gateways**，这个文件的目的是：提供网关的静态信息，“预先载入”到 **routed** 的路由表里。

13.5 gated：更好的路由守护进程

gated 是一种不但好而且一度可以免费获得的路由构架，它允许同时使用许多不同的路由协议。**gated** 对被公告的路由、广播地址、信任策略和代价度量提供了精确的控制。它能够在几种协议之间共享路由，它能够在不同路由系统上标准化的区域之间构建路由网关。**gated** 是有着最好的管理接口和配置文件设计的 Linux 管理软件之一。

gated 一开始是可以自由发布的软件，但在 1992 年它变成了私有的软件，并转交给了 Merit GateD 联盟（Merit GateD Consortium）。**gated** 的商业版本只提供给该联盟的成员。这个联盟最后解体了，NextHop 取得了 **gated** 的所有权，NextHop 是一家嵌入式网络软件开发公司。这次变迁当即结束了 **gated** 在开放源代码世界中的生涯，身后留下了一个个苦涩的故事。

XORP（eXtensible Open Router Platform，扩展的开放路由器平台）是一个很有希望的新项目，它的涌现有助于填补 **gated** 陨落后留下的空白。虽然 XORP 现在还只处于 beta 测试阶段，但它已经为生产环境的使用做好了准备，我们希望它能成长到代替 **gated** 的前任角色。在 www.xorp.org 可以了解到最新进展。

同时，需要多协议路由的地方可以考虑使用 GNU Zebra（www.zebra.org），这是一个能在大多数 Linux 平台上运行的全面的路由软件包。遗憾的是，它缺乏在一个生产环境中管理动态路由选择所需的大多数功能、方便性和详细文档。在这种情况下，要最大限度地利用好您的资源，需要购买一台专业路由器（比如 Juniper 或者 Cisco 的产品）。

13.6 路由策略的选择标准

在管理网络路由上，按复杂性基本上可分为 4 个级别：

- 无路由；
- 只有静态路由；
- 大部分为静态路由，但客户机监听 RIP 更新；
- 到处都有动态路由。

整个网络的拓扑结构对每一网段的路由需求都有极大的影响。不同的网络可能需要的路由支持水平也不同。下面的规则能够帮助您选择策略。

- 独立的网络不需要路由。
- 如果只有一条通向网络之外的路径，那么网络上的客户机（非网关机器）应该有一条通向唯一网关的静态默认路由。除了网关自身外，不需要其他配置。
- 如果一端是和少数几个网络连接的一个网关，另一端是通往外部世界的一个网关，那么有多条显式指向前者的静态路由和一条指向后者的默认路由。然而，如果两端都有多种路由选择，那么动态路由更可取。
- 如果使用 RIP 并在乎它所造成的网络和系统的负载，那么应当避免以主动模式运行 **routed**——它会以很短的时间间隔广播它自己所知道的一切东西（不管正确与否）。用 **gated -q** 可以让客户机被动监听路由更新，而不会把它们自己的信息发送出去。

www.TopSage.com

- 许多人都会告诉您 RIP 是一种令人恐怖的协议，而 **routed** 是撒旦的化身。但情况不一定如此。如果 RIP 对您适用而且您也满意它的性能，那么可以继续使用它。如果您把时间花费在设计过分的路由策略上，则会得不偿失。
- **routed** 监听每一项信息而且相信它所听到的一切事情。即使站点使用 RIP，也可能想使用一台专用路由器（比如一台 Cisco 路由器）来管理路由数据的交换，而只在客户机上运行 **routed**。
- 在网络跨越政治或管理界限的地方使用动态路由。
- 如果可能的话，在包含某些环路或冗余路径的动态路由网络上使用 OSPF。
- 通过多家上游提供商连到 Internet 的路由器必须使用 BGP。不过，连到 Internet 的许多路由器都只有一条上游路由，因此只要一条简单的静态路由就足够了。

对于中等规模的网络（具有相对稳定的本地结构，与别人的网络有一条连接）而言，比较好的路由策略就是组合使用静态和动态路由。在没有外网网关的本地结构中，机器可以使用静态路由，把所有不能被识别的包转发到默认机器上，而且这台默认路由主机知道外部世界，它使用动态路由。

如果网络太复杂，不能用这种方案管理，则应当求助于动态路由。在叶子（leaf）网络上仍然使用默认静态路由，但有多台路由器的网络上的机器应该以被动模式运行 **routed**。

13.7 Cisco 路由器

当今，Cisco 公司制造的路由器是 Internet 路由选择的事实标准。Cisco 公司的产品很著名，占有路由器市场 70%的份额，并且也比较容易能找到知道怎样运行 Cisco 产品的人员。在有 Cisco 之前，常把有多个网络接口的 UNIX 机器当路由器来用。今天，人们喜欢使用专用路由器，它们可以放在数据通信室和天花板上，在那儿和网线放到一起。与使用 UNIX 或者 Linux 的路由器相比，专用路由器更便宜、更快并且更安全。

多数 Cisco 路由器产品运行 Cisco IOS 操作系统，它是专有操作系统，与 Linux 无关。它的命令集相当大，全部文献加起来会占 4.5 英尺长的书架。我们肯定不会在此详细介绍 Cisco IOS，但知道一点基本知识还是有好处的。

IOS 定义了两级访问权限（用户级和特权级），每一级都有保护口令。默认情况下，您可以 **telnet** 到一个 Cisco 路由器，进入用户模式[4]。系统会提醒您输入用户级的访问口令：

```
$ telnet acme-gw.acme.com
Connected to acme-gw.acme.com.
Escape character is '^]'.

User Access Verification
Password:
```

输入了正确的口令之后，您就会得到 Cisco 的 EXEC 命令解释器的提示符。

```
acme-gw.acme.com>
```

在这个提示下，您可以输入 **show interfaces** 这样的命令来查看路由器的网络接口，或者输入 **show ?** 这样的命令获得帮助。

若要进入特权模式，输入 **enable**，然后在要求输入口令的时候输入特权口令。一旦您进入了特权模式，提示符以#号结尾：

```
acme-gw.acme.com#
```

[4] 可以配置多种访问方法。如果网点已经使用了 Cisco 路由器，可以和网络管理员联系，了解可以使用哪些方法。

www.TopSage.com

注意，在这个提示符下，您什么都可以做，包括删除路由器的配置信息和它的操作系统。在拿不准的时候，请参阅 Cisco 手册或任何一本由 Cisco Press 出版的综合书籍。

输入 **show running** 可查看路由器的当前运行配置，输入 **show config** 可查看当前的固定配置。在大多数的情况下，这二者是相同的。

以下是典型的配置：

```
acme-gw.acme.com# show running
Current configuration:
version 12.1
hostname acme-gw
enable secret xxxxxxxx
ip subnet-zero

interface Ethernet0
description Acme internal network
ip address 192.108.21.254 255.255.255.0
no ip directed-broadcast
interface Ethernet1
description Acme backbone network
ip address 192.225.33.254 255.255.255.0
no ip directed-broadcast

ip classless
line con 0
transport input none

line aux 0
transport input telnet
line vty 0 4
password xxxxxxxx
login

end
```

修改路由器配置的方法各种各样。Cisco 提供了一些图形工具，可以在 UNIX/Linux 和 Windows 的某些版本下运行。真正的网络管理员从来不使用那些工具，因为命令行总是比较可靠的。您可以用 **tftp** 上传一个配置文件到路由器，或者用 **tftp** 从路由器下载配置文件，这样就能用自己喜欢的编辑器来编辑它[5]。

为了修改来自命令行的配置，可以键入 **config term**：

```
acme-gw.acme.com# config term
Enter configuration commands, one per line.  End with CNTL/Z.
acme-gw(config)#
```

然后，您便能够输入新的配置命令以让它们可以按照您的想法输出。例如，如果想改变上述例子中 Ethernet0 接口的 IP 地址，那么可以输入：

```
interface Ethernet0
ip address 192.225.40.253 255.255.255.0
```

在输完了配置命令之后，请按<Control-Z>返回到正常的命令提示。如果您对新配置很满意，那么输入 **write mem** 把配置保存到固定内存。

下面是成功配置 Cisco 路由器的经验技巧。

[5] 重点提示：Microsoft Word 可不是这个应用的最好选择。

www.TopSage.com

- 用 **hostname** 命令命名路由器。这一预防措施有助于防止因为在错误的路由器上改变配置而引起的意外情况。主机名总是应该出现在命令提示符中。
- 手头应当总是保留一个路由器配置的备份。您可以写一个简短的 **except** 脚本，它每晚把运行配置用 **tftps** 下载到一台 Linux 机器，以安全保存。
- 通过在路由器的 VTY（VTY 类似于 UNIX 上的 PTY）上配置访问列表来控制对路由器命令行的访问。这一预防措施可以避免未经授权的一方侵入您的路由器[6]。
- 在每个接口上使用访问列表，控制网络之间（可能还有到外界）的通信流量。参考 20.12 节，了解设置访问列表的更多信息。
- 保持路由器的物理安全性。如果可以在物理上接触到 Cisco 机器，很容易就能重置特权口令。

13.8 推荐读物

PERLMAN, RADIA. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols (2nd Edition)*. Reading, MA: Addison-Wesley. 2000.

这是本章所讨论主题的权威著作。如果您只买一本讲网络基础的书，那么就应该买这本。另外，不要错过和 Radia 在一起的机会——她是个非常风趣的人，在她的脑子里有令人惊异的渊博知识。

HUITEMA, CHRISTIAN. *Routing in the Internet, Second Edition.* Prentice Hall. 2000.

这本书从基础开始清楚详细地介绍了路由技术。它涵盖了常用的大多数协议，还包括一些诸如多播这样的高级主题。

MOY, JOHN T. OSPF: *Anatomy of an Internet Routing Protocol.* Addison-Wesley. 1998.

OSPF 协议标准的制订者对 OSPF 的完整说明。

STEWART, JOHN W. BGP4 *Inter-domain Routing in the Internet.* Addison-Wesley. 1999.

还有许多与路由技术相关的 RFC。表 13.3 列出了主要的一些 RFC。

表 13.3 与路由技术有关的 RFC

RFC	名 称	作 者
2328	OSPF Version 2	John T. Moy
1058	Routing Information Protocol	C. Hedrick
2453	RIP Version 2	Gary Scott Malkin
1256	ICMP Router Discovery Messages	Stephen E. Deering
1142	OSI IS-IS Intra-domain Routing Protocol	David R. Oran
1075	Distance Vector Multicast Routing Protocol	D. Waitzman 等
4632	CIDR: an Address Assignment and Aggregation Strategy	Vince Fuller 等
4271	A Border Gateway Protocol 4 (BGP-4)	Yakov Rekhter 等

13.9 习题

E13.1 研究 Linux 的 **route** 命令，简要介绍它的功能。使用 **route** 命令，怎样做到：

[6] 最新的 IOS 版本支持 SSH 协议。如果您的环境里有它，应该用它来代替标准的 TELNET 接口。

www.TopSage.com

a）加一条通过接口 eth1 到 128.138.129.1 的默认路由；

b）删除到 128.138.129.1 的默认路由；

c）判断一个程序（**routed**）或者一个 ICMP 重定向是否加入了一条路由（注意，这个方法也能用 **netstat -rn** 命令的输出做到）。

E13.2 比较静态路由和动态路由，举出彼此的优缺点。分别介绍两者适用的几种场合，说明原因。

★E13.3 考虑下面的 **netstat -rn** 输出。介绍各条路由，推断网络的设置。10.0.0.0 还是 10.1.1.0 更靠近 Internet？每条路由是由哪个进程加的？

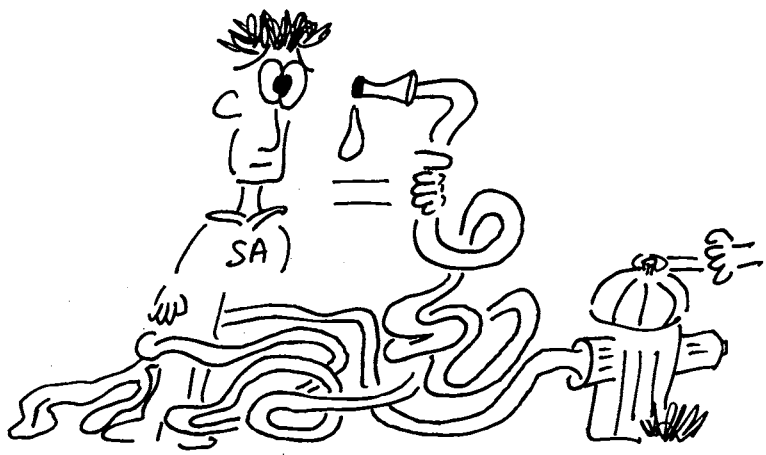
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	40	0	0	eth1
10.1.1.0	0.0.0.0	255.255.255.0	U	40	0	0	eth0
0.0.0.0	10.0.0.1	0.0.0.0	UG	40	0	0	eth1

★★E13.4 推断您的网点所使用的路由方案。采用的协议是什么？哪些机器连接到了 Internet？您可以使用 **tcpdump** 来寻找本地网络上的路由更新包，使用 **traceroute** 探测本地网络之外的网络（需要 root 权限）。

★★E13.5 如果您是一个中型 ISP，提供拨入账号和虚拟主机业务，您会采用哪种路由选择的设置？要保证不但考虑到了 Internet 主干和您自己的网络之间的网关路由器，而且还考虑到了可能使用的任何内部路由器。画一张网络图，勾勒出您的路由结构。

www.TopSage.com

第 14 章　网络硬件



无论是从世界各地传来的视频图像，还是您的孩子从楼下传来的声音，我们生活的这个世界里每样东西都是以数字形式处理的。数据可以从一个地方飞快传送到另一个地方则是尽人皆知的事情。在这种疯狂现象的背后，有着奇异的网络硬件以及——如您猜测的那样——来自深邃、隐秘的 UNIX 宝库中的一整套知识。如果说 UNIX 技术已经在一个领域深入了人类的生活，那么这个领域就是大规模实现了分组交换形式的数据传输方式。

跟得上所有这些快速传送的比特位颇具挑战。您的网络速度和可靠性当然会对您的机构效率有直接的影响，但是现如今的网络技术如此普及，网络的状态都会影响到我们与他人的许多基本交往能力（比如打个电话）。设计拙劣的网络会让个人和专业人员身处窘境。修补这个网络的花费也是很昂贵的。

成功的网络安装至少要归功于 4 个主要的因素：

- 形成一种合理的网络设计；
- 选择高质量的硬件；
- 正确的安装和建档；
- 良好的运行和维护。

本章的第 1 节讨论一些常用于局域网和广域网（包括以太网、ATM、帧中继和 DSL）的介质。然后介绍您在任何网络中可能面对的设计问题，其中有新问题，也有老问题。

14.1　选择 LAN、WAN、还是 MAN

从某种意义上讲，我们是幸运的，因为 TCP/IP 很容易就能通过各种各样的介质进行传输。不过，网络硬件市场实际上已经分成了各种混乱的门类。

www.TopSage.com

在一座建筑物或一群建筑物之内存在的网络通常称为局域网（Local Area Network），或者缩写为LAN。局域网中盛行高速、低成本的连接。在广域网（Wide Area Network，WAN）中，通信端点的地理位置是分散的，可能相隔数千公里。在这些网络上，高速度通常会带来高成本，但实际上，网络上可以包含的站点并没有界限（从比利时的布鲁日到阿拉斯加的西卡）。MAN是一个电信业的营销词汇，代表Metropolitan Area Network（城域网），它意味着在一个城市或一些城市范围内使用的高速、成本适中的访问介质。在这一章里，我们研究用于实现这些网络的一些技术。

14.2 以太网：常见的局域网

以太网已经占据了世界范围内局域网市场90%以上的份额，到处都可以见到多种形式的以太网。它首先出现在MIT（麻省理工学院）Bob Metcalfe的博士论文中。Bob毕业后进入Xerox PARC公司。Xerox公司与DEC和Intel公司一起最终将以太网开发成一个产品。它是相互竞争的计算机公司联合起来完成一个技术项目的首批案例之一[1]。

以太网最初规定为3Mbit/s（兆比特每秒），但几乎立即就变成了10Mbit/s。1994年，随着以太网有了100Mbit/s的标准，它获得了人们的注意。到1998年，在以太网19岁的时候，它准备好了在1Gbit/s上打一场新的战争。现在，20多岁成熟的以太网已经超过了它所有的竞争对手，并在光纤上实现了向10Gbit/s以太网的迈进。表14.1显示了各种以太网标准的发展历程[2]。

表14.1 以太网的演变

年 代	速 率	通 用 名 称	IEEE编号	距 离	介 质
1973	3 Mbit/s	Xerox以太网	–	?	同轴电缆
1976	10 Mbit/s	Ethernet 1	–	500m	RG-11同轴电缆
1982	10 Mbit/s	DIX以太网 （以太网II）	–	500m	RG-11同轴电缆
1985	10 Mbit/s	10Base5（“粗缆”）	802.3	500m	RG-11同轴电缆
1985	10 Mbit/s	10Base2（“细缆”）	802.3	180m	RG-58同轴电缆
1989	10 Mbit/s	10BaseT	802.3	100m	3类UTP铜导线
1993	10 Mbit/s	10BaseF	802.3	2km 25km	多模光纤 单模光纤
1994	100 Mbit/s	100BaseTX（“百兆”）	802.3u	100m	5类UTP铜导线
1994	100 Mbit/s	100BaseFX	802.3u	2km 20km	多模光纤 单模光纤
1998	1 Gbit/s	1000BaseSX	802.3z	260m 550m	62.5μm多模光纤 50μm多模光纤
1998	1 Gbit/s	1000BaseLX	802.3z	440m 550m 3km	62.5μm多模光纤 50μm多模光纤 单模光纤
1998	1 Gbit/s	1000BaseCX	802.3z	25m	双绞线
1999	1 Gbit/s	1000BaseT（“千兆”）	802.3ab	100m	5E类和6类UTP铜导线
2002	10Gbit/s	10GBase-SR 10GBase-LR（“万兆”）	802.3ae	300m 40km	多模光纤 单模光纤
2006	10Gbit/s	10GBase-T	802.3an	100m	7类UTP铜导线

[1] Bob Metcalfe还提出了“Metcalfe定律”，即网络的价值将会随着用户数量的增加而呈指数增长。

[2] 我们已经略去了几种经证明没有得到普遍使用的以太网标准，比如100BaseT4和100BaseVG-AnyLAN。

www.TopSage.com

续表

年 代	速 率	通 用 名 称	IEEE 编号	距 离	介 质[a]
2006[b]	100Gbit/s	待定	待定	待定	光纤
2008[b]	1Tbit/s	待定	待定	待定	CWDM 光纤
2010[b]	10Tbit/s	待定	待定	待定	CWDM 光纤

a. MM=多模，SM=单模，UTP=无屏蔽双绞线，CWDM=粗波分复用，DWDM=密集波分复用。
b. 业界预测。

14.2.1 以太网的工作方式

我们可以把以太网描述为一个彬彬有礼的晚宴，在晚宴上，要说话的客人（计算机）并不会打断别人，而是在开口说话之前等待谈话安静下来（在网络电缆上没有通信流量）。如果两位客人同时开始说话（冲突），那么他们都会停下来，互相道歉，等上一会儿，然后他们其中的某一位再开始说话。

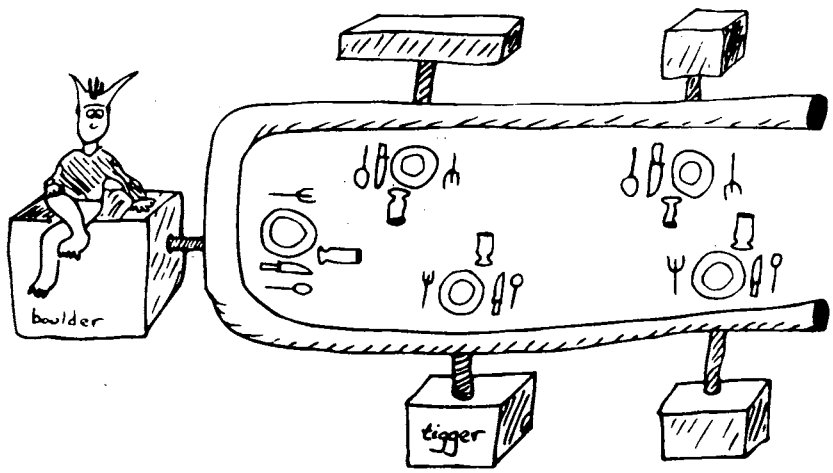


图 14.1 彬彬有礼的以太网晚宴

这个方案的技术术语就是 CSMA/CD。

- 载波侦听（Carrier Sense）：您能够分辨出是否有人正在讲话。
- 多路访问（Multiple Access）：每个人都能够讲话。
- 冲突检测（Collision Detection）：您知道您在什么时候打断了别人的讲话。

实际延迟要根据冲突检测进行，它有点随机性。这种约定能避开这样的情况：两台主机同时向网络发送信号、检测冲突、等待相同的时间，然后再开始发送信号，这样网络上就充满了冲突。但实际情况绝不能是这样！

14.2.2 以太网拓扑结构

以太网拓扑结构是没有环路的分支总线结构，同一个网络上任意两台主机之间只有一条让包通过的路径。以太网在一个网段上交换 3 类包：单播（unicast）、多播（multicast）和广播（broadcast）。单播包只被传送到一个主机，多播包被传送到一组主机，广播包被传送到一个网段上的所有主机。

“广播域（broadcast domain）”是接收发往硬件广播地址的包的主机集合，每个逻辑上的以太网段只有一个广播域。在早期的以太网标准和介质下（例如 10Base5），物理段和逻辑段实际上是一样的，

www.TopSage.com

因为所有的包都在一条大电缆上传送，主机接口都绑在电缆的一侧[3]。

随着现代交换机的出现，现在的逻辑网段通常由许多（可能是几十或几百个）物理网段（或者在有些情况下是无线网段）构成，而每个物理网段上面只连接两个设备：交换机端口和主机。交换机负责把单播和多播包发送到接收方所在的物理（无线）网段上；广播流量则被转发到一个逻辑网段的所有端口上。

一个逻辑网段可能包含了以不同速度（10Mbit/s、100Mbit/s、1Gbit/s 或 10Gbit/s）运行的物理网段（或者无线网段），因此，为了消除潜在的冲突，交换机必须有缓冲和计时功能。

14.2.3 无屏蔽双绞线

对以太网而言，无屏蔽双绞线（UTP，Unshielded Twisted Pair）是首选的电缆介质。它以星型拓扑结构为基础，与其他介质相比，它有几个优势：

- 它使用便宜的、易于获得的铜导线（有时也能使用现成的电话线）；
- UTP 导线比同轴电缆或光纤更容易安装和调试，也更容易定制长度；
- UTP 使用 RJ-45 连接器，这种连接器便宜、可靠而且容易安装；
- 到每个机器上的链路是独立的（独占的），所以在一条链路上的硬件故障或电缆问题不太可能影响网络上的其他主机。

图 14.2 展示了 UTP 网络通常的“样子”。

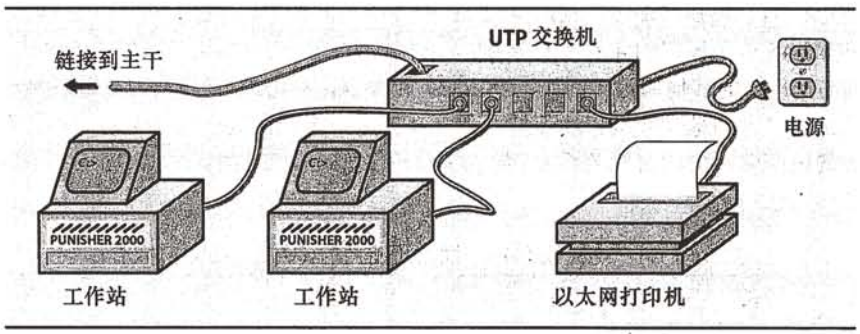


图 14.2 UTP 的安装

我们把适合现代局域网使用的 UTP 导线划分成 8 类。一家大型的电缆供应商 Anixter 首先引入了按性能定级的体系。TIA（Telecommunications Industry Association，电信行业协会）正式规定了这些标准，就是现如今的 1 类到 7 类线，其间有一个特殊的 5E 类线。

国际标准化组织（International Organization for Standardzation，ISO）也已经投身于给电缆分类这一刺激而又利润颇丰的领域之中了，该组织推出的标准和 TIA 的高标号分类正好或者近似等价。例如，TIA 的 5 类线等价于 ISO 的 D 类电缆。对于本书读者里的计算机迷们来说，表 14.2 给出了现在各种分类体系之间的主要区别。牢记这些知识，您就可以在朋友面前露一手了。

实际上，1 类和 2 类电缆线仅仅适合于语音应用（如果说适合用的话）。3 类线的速度是可以用于局域网的最低速度，它是 10Mbit/s 10BaseT 的标准。4 类线有点孤立，它并不确切适用于哪种特殊的应用。它偶尔用于安装 16 Mbit/s UTP 令牌环或特殊的 10BaseT。5 类线支持 100Mbit/s，而且是目前数据接线中最常用的标准。5E 类和 6 类线支持 1Gbit/s。在铜导线上的 10Gbit/s 以太网标准出台之后，7

[3] 我们不是在开玩笑！要连接一台新的计算机，就要用一种特殊的钻在电缆外包皮上钻一个洞深入到中心处的导线。连接外部导线的“钻入式接头”要使用螺丝拧紧。

www.TopSage.com

类线就用于 10Gbit/s 以太网。

表 14.2 UTP 电缆的特性

参 数[a]	5 类 D 型[b]	5E 类	6 类 E 型	7 类 F 型
频率范围	100 MHz	100 MHz	250 MHz	600 MHz
衰减	24 dB	24 dB	21.7 dB	20.8 dB
NEXT	27.1 dB	30.1 dB	39.9 dB	62.1 dB
ACR	3.1 dB	6.1 dB	18.2 dB	41.3 dB
ELFEXT	17 dB	17.4 dB	23.2 dB	?[c]
回波损耗	8 dB	10 dB	12 dB	14.1 dB
传输延迟	548 ns	548 ns	548 ns	504 ns

a. NEXT=近端串扰，ACR=衰减-串扰比，ELFEXT=等效远端串扰。
b. 分别包括 TIA 和 ISO 增加的 TSB95 和 FDAM2 要求。
c. 这个参数目前不确定，留给以后研究。

10BaseT 连线需要两对 3 类线，而且每条链路的长度限制为 100 米；100BaseTX 的长度限制也是 100 米，但它需要两对 5 类线。既有以 PVC 包裹的导线，也有以特氟纶（Teflon）包裹的导线。选择导线外包皮应该与安装电缆的环境相适应。装进建筑物通风系统（回风区）的导线一般需要使用特氟纶[4]。与特氟纶相比，PVC 更便宜且更容易使用。关于布线的更多信息请参考 14.11 节。

RJ-45 连接器使用 1、2、3 和 6 针来连线。虽然速度为 10Mbit/s 或 100Mbit/s 的连接只需要两对导线，但我们建议：当安装新网络时，使用 4 对 5E 类导线，并把 RJ-45 插口的 8 个引线都连接起来。

接线板和 RJ-45 墙上插口的 4 对 UTP 导线端的接线方法，建议使用 TIA/EIA-568A RJ-45 接线标准。这个标准与 RJ-45 的其他用途（例如，RS-232）相兼容，不论是否能够轻易地接触到电缆对本身，使用这个标准都能很方便地使两端接线保持一致（译者注：作者在这里是想说明，不必像平常那样沿着导线查才能找出两端对应的接线关系）。表 14.3 给出了 568A 标准的详细信息。关于 RS-232 标准的更多信息，请参考 27.1 节。

表 14.3 把 4 对 UTP 连接到 RJ-45 插口所使用的 TIA/EIA-568A 标准

线 对	颜 色	连 到	线 对	颜 色	连 到
1	白色/蓝色	5/4 针	3	白色/绿色	1/2 针
2	白色/橙色	3/6 针	4	白色/棕色	7/8 针

建筑物内已经布好的线可能适合也可能不适合网络使用，这取决于导线的安装方式和安装时间。在 20 世纪 50 年代和 60 年代，许多旧楼都得到翻新，安装了新电缆。遗憾的是，这些电缆一般连 10Mbit/s 的速度都不支持。

14.2.4 连接和扩展以太网

从逻辑上来说，以太网实现了在 ISO 网络 7 层模型中几个地方的连接。在第 1 层，也就是物理层上，使用的要么是硬件连接器，要么是中继器（现在通常称为集线器）。它们直接传送信号，就像是用绳子连接两听罐头。

在第 2 层，也就是数据链路层上，使用的是交换机。交换机基于硬件的源和目标地址传送帧，就像只读瓶子外边的标签来发送瓶子里的一条消息一样。

[4] 您可以和消防队长或者当地的消防部门协商一下，以确定本地的要求。

www.TopSage.com

在第 3 层，也就是网络层上，使用的是路由器。路由器根据最终接收方的位置把消息传送到下一跳，类似于查看瓶子里的消息以确定消息究竟是发送给谁的。

一、集线器

集线器（也称为集中器）是连接 UTP 以太网络的物理网段的主动设备。它们需要外接电源。作为中继器，集线器重新定时和重构以太网帧，但不解释帧。它不知道包将要发送到什么地方，也不知道包使用的是什么协议。

网络上两个最远点之间一定不要超过 4 个集线器。第一版和第二版的以太网指定每个网络上最多串联 2 个集线器。IEEE 802.3 标准把 10Mbit/s 以太网的限制拓宽到 4 个。100Mbit/s 以太网允许串联 2 个中继器，1000BaseT 以太网只允许使用一个，而 10Gbit/s 的以太网根本不允许有串联。图 14.3 给出了 10Mbit/s 网络的合法和不合法的配置。

系统管理员偶尔也需要检查集线器，所以不应该把它们放置在暗处或很难接触到的地方。如果它们出现问题，只要重新加电即可恢复。

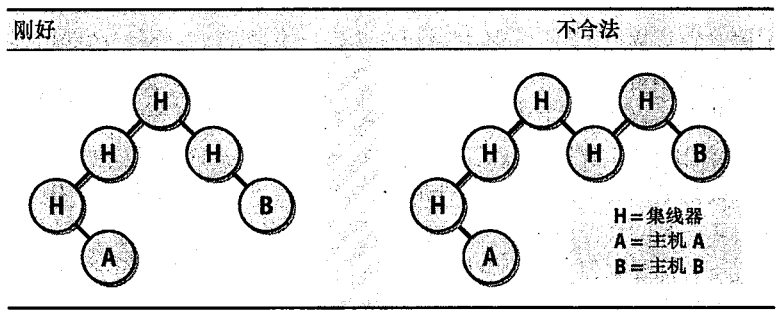


图 14.3 计算集线器的个数

二、交换机

交换机在 ISO 模型的数据链路层（第 2 层）连接以太网。它们的目的是用某种方法把两个不同的物理网络连接起来，让这两个物理网络看起来就像是一个大的物理网络。现在，交换机是连接以太网设备的业界标准。

交换机用硬件来接收、再次产生和再次发送包[5]。大多数交换机使用动态学习算法。它们会注意哪个源地址从这个端口来，而哪个源地址从那个端口来。只有需要的时候才在端口之间转发包。一开始交换机会转发所有包，但几秒之后，交换机了解到更多主机的位置，转发就有了更强的选择性。

因为并不是所有包都要在网络之间转发，所以每段电缆上的通信量就比所有机器都在同一条电缆上的通信量要小。因为大多数通信都倾向于在本地进行，所以表面上带宽的增加是很惊人的。而且，因为网络的逻辑模型不会因为有交换机出现而受到影响，所以安装交换机几乎对管理没有什么影响。

如果网络包含环路，那么交换机有时会被搞糊涂，因为从一台主机来的包会出现在交换机的两个（或更多）端口上。单独一个以太网不可能有环路，但在用路由器和交换机把几个以太网连在一起时，拓扑结构可能会包含到一台主机的多条路由。针对这种情况，有些交换机能保留另一条逆向路由，以防主路由故障。交换机对它们看到的网络进行修剪，直到在网络上剩下的部分中到每个节点只有一条路由为止。有些交换机也能够处理两个相同网络之间的多条链路，而且用一种轮流的方式路由流量。

因为交换机固件中内置了更多功能，所以交换机在不断变得“聪明”起来。某些交换机可以用来监视网络安全。它们把自己所看到的所有陌生的以太网地址都记录下来，据此来检测和汇报新连接的机器。因为交换机是在以太网层运行的，所以交换机与协议无关，能够处理高层包类型（例如，IP、AppleTalk 或 NetBEUI）的随意混用。

[5] 因为要再次生成包，并对它重新定时，所以全交换网络不会受到图 14.3 所示的“中继器数量”的限制。

www.TopSage.com

交换机必须扫描每个包以确定是否应该转发这个包。通常用包扫描速率和包转发速率来衡量交换机的性能。许多厂商在它们提供的性能说明中没有提到包的大小，因此，实际性能没有广告中宣传的那么好。

虽然以太网的交换硬件一直都在变得更快，但如果要在一个单独的逻辑网段上连接 100 多台主机，那么它仍然不是合理的技术。大型交换式网络经常遭受“广播风暴”这样的问题，原因是广播流量必须要转发到一个交换式网段的所有端口上。为了解决这个问题，应当使用路由器来隔离交换式网段之间的广播流量，从而创建多个逻辑以太网。

大型网点会得益于那些能够把它们的端口分成（通过软件配置）若干称为 VLAN（Virtual Local Area Network，虚拟局域网）的子组的交换机。VLAN 是属于同一个逻辑网段的一组端口，就好像这些端口是连接在自己专用的交换机上一样。这样的划分增加了交换机隔离流量的能力，而且对安全性和性能都产生了有益的效果。

VLAN 之间的流量由一个路由器来处理，或者在某些情况下是一个路由模块或者交换机中的路由软件层来处理。这种系统有一种叫做“VLAN Trunking（VLAN 聚合）”（就像 IEEE 802.1Q 协议提供的功能那样）的扩展，它从物理上分隔交换机，以便为同一个逻辑 VLAN 上的端口提供服务。

选择交换机可能比较困难。在计算机行业中，交换机市场的竞争很激烈，而且它还受到了那些甚至没有半点真实可言的市场宣传的影响。为了购买交换机而选择一家交换机生产厂商的时候，您应该依靠独立的评测（“烘烤比赛”，就像杂志里出现的那些产品对比），而不是根据厂商提供的任何数据。最近几年有一种现象很常见：某一厂商在几个月的时间里拥有“最好”的产品，但当它想对产品进行改进时，就会彻底毁坏产品的性能或可靠性，这样就使另一家厂商处于顶峰时期。

无论在什么情况下，都要确保交换机背板有足够的速度——这是在很长一段时间之后仍然真正有价值的数字。设计良好的交换机的背板速度应该超过它所有端口的速度之和。

三、路由器

路由器是专用计算机，它包含两个或多个网络接口，并在 ISO 协议栈的第 3 层（网络层）指挥流量。路由器以 TCP/IP 协议包头的信息为依据，把包来回转发到它们的最终目标。除了把包从一个地方传送到另一个地方的简单功能之外，它们还能执行其他功能，例如包过滤（出于安全原因）、确定优先次序（出于服务质量的原因）、发现网络整体拓扑结构。有关路由技术的工作原理，请参考第 13 章。

我们能够在一个路由器上找到许多不同类型的硬件接口（例如 SONET、以太网和 ATM）。就软件方面而言，一些路由器也能够处理像 IPX 或 AppleTalk 这样的非 IP 流量。在这些配置中，必须为您想处理的每种协议配置路由器和它的接口。现如今，总的来说要远离这些老协议，好好支持 TCP/IP，这才是个好想法。

路由器采取的形式有两种：固定配置和模块。固定配置路由器的特定网络接口在工厂里就永久地安装好了。它们通常适合于小型的特殊应用。例如，如果要把一个小公司连接到 Internet 上，那么使用有一个 T1 和一个以太网接口的路由器就是一个很好的选择。

模块化的路由器有一个插槽式或总线式的结构，终端用户能够把接口添加到这个结构上。虽然这种方法通常要贵一些，但它的灵活性更强。

根据可靠性的需要和预计的流量负载，专用路由器可能比配置成路由器的 Linux 系统便宜，也可能更贵。不过，一般来说，专用路由器将提供更优良的性能和可靠性。这是网络设计的领域之一，为了避免以后的麻烦，通常合理的做法应该是在这上面预先多花点儿钱。

14.3 无线：流动人士的 LAN

无线连网是一个增长很快的热门领域，现在生产级（production-grade）产品的价格已经能让大家

www.TopSage.com

买得起了。对比最近在有线网络技术上取得的进步，无线网络的速度（通常介于 2Mbit/s 到 54Mbit/s 之间）似乎有点儿不能满足企业应用的需要。实际上，对于很多用途来说，这些速度还是足够了。在家用或者小型办公环境里的 11Mbit/s 无线网络就是系统管理员的梦想。到了 54Mbit/s 的速度，无线网络也能为企业环境所接受了。此外，用于商业展览、咖啡厅、码头、机场和其他公共场所的无线接入技术，对于许多人来说，也确实从遥不可及的时代转入了超级连接的时代。

无线连网技术最有希望的标准是 IEEE 的 802.11g 和 802.11a 无线局域网标准。802.11g 工作在 2.4GHz 频段，它以 54Mbit/s 的速率提供了类似 LAN 的接入方式。随设备和地形的不同，其运行距离从 100 米到 40 公里不等。802.11a 也提供了高达 54Mbit/s 的带宽，但使用了 5.4GHz 的频段。当前有些设备把两个通道合到一起提供 108Mbit/s 的带宽。

虽然 802.11g 和 802.11a 都宣称运行在 54Mbit/s 下，但是它们的目标和实现的带宽却大相径庭。802.11g 主要面向消费市场，它一般比 802.11a 便宜，提供 3 个不重叠的数据通道。与此对照，802.11a 有 12 个不重叠的数据通道。通道非常像高速公路上的车道：通道越多，获得全速带宽的客户机数量也越多。

对于小型网点来说，无论哪一种标准可能都可以。部署比较大的网点或者园区网就要考虑 802.11a 了，因为频道数量很大。在实现上，目前的大多数无线发射设备两种网络都能用。

如今 802.11b（11Mbit/s）、802.11g 和 802.11a 的网络都已经变得司空见惯了。无线网卡价格便宜，而且大多数膝上机和桌面 PC 都可以用（或者干脆就直接内置了）。和有线以太网一样，802.11 网络最常用的结构也是用一个中继器（按无线的说法称为“接入点[access point]”）当作多个客户机的连接点。接入点可以连接到传统的有线网络上，也可以以无线方式连接到其他接入点上（这种配置称为“无线网状网[wireless mesh]”）。

如果您有合适的硬件和驱动程序，就可以把一台 Linux 机器配置成一个 802.11a/b AP。我们至少知道有一种芯片支持这项功能，这种芯片是 Intersil Prism II。家用或者小型办公室用的独立 802.11a/b 无线基站，比较好的有 Apple 公司的 AirPort Express，它是一种外观像个电源插头的产品，价钱不贵（大约 150 美元），而且功能不错[6]。马上买一个试试吧！

数十家厂商都在销售无线 AP 产品。您可以从家得宝（译者注：Home Depot 是全球最大的家具建材零售商）公司乃至杂货店买到它们。不出我们所料，“一分钱一分货”的道理也同样适用。便宜的 AP（在 50 美元价位内）在处理大文件传输或者多个活动客户机的时候可能性能很差。

调试一个无线网络也是有点儿没有准头的事儿，因为在出现问题的时候，其中会有大量的不定因素存在。如果您在一个企业规模的环境里部署无线网络，那么或许需要投资购买一套无线网络分析器。我们高度推荐 AirMagnet 公司出品的分析产品。

14.3.1 无线网络的安全

目前，无线网络的安全性还很差。WEP（Wired Equivalent Privacy）是一种配合 802.11b 网络使用的协议，它对无线传输的包提供 40 位、104 位或者 128 位加密。遗憾的是，这个标准在设计上有一个致命缺陷，对于偷听者来说，它比一个睡着的警察（译者注：speed bump 的原意指道路上的减速带，引申指睡着的警察，让车速降低但不会被抓）强不了多少。有些坐在您的建筑物之外的人可以直接访问到您的网络，而且不会被觉察到。

最近，WPA（WiFi Protected Access）安全标准的颁布增强了人们对于无线网络安全的信心。现在应该在所有新安装的无线网络上使用 WPA 来代替 WEP。没有 WPA 的无线网络——不管有没有 WEP——都应该认为完全不安全。

[6] 实际上，它也能插到立体声音响里，以无线方式播放 PC 或者台式机上的音乐。

www.TopSage.com

802.11i，也称为 WPA2 是更新的代替 WPA 的安全标准。它增加了针对企业无线网络的更多身份认证机制。

14.3.2 无线交换机

就像以太网的集线器发展为以太网交换机一样，为了在大型企业中使用，无线产品也正在经历一个逐渐翻新的过程。许多厂商（比如 Airespace）现在生产“无线交换机”，配合部署在园区内的一片 AP 工作。其背后遵循的理论是，分散布置廉价的大批 AP，然后让一个“智能”交换机集中管理它们。交换机维护着 WAP 的配置信息，平稳地支持身份验证和漫游功能。

如果您需要在一个中到大型的机构里提供普遍的无线服务，那么肯定要花时间评估一下这类产品。不仅要降低花在管理上的时间，而且最为重要的是，要包括一种监视和管理面向用户的 QoS（服务质量，quality of service）的手段。

在您的整个环境里部署一个 802.11a/b/g 网络，用它支持给员工配备的手持式 VoIP 电话，这可以算是个独特的诀窍。它就像是个免费的移动通信网络。

14.4 FDDI：令人失望且昂贵的局域网

在 10Mbit/s 的速度上，20 世纪 80 年代的以太网没有为某些网络需求（例如通过公司或园区主干连接工作组）提供足够的带宽。为了提供可以选用的更高带宽，ANSI X3T9.5 委员会制订了 FDDI（Fiber Distributed Data Interface，光纤分布式数据接口）标准，并把其作为以太网的备选标准[7]。FDDI 作为 100Mbit/s 速度的令牌环网来设计和销售，这使 FDDI 一度看起来似乎就是解决许多机构的带宽需求的便利方案。

遗憾的是，FDDI 绝对在每个方面一直令人失望，现在不应该考虑把它用在生产目的上。我们之所以在这里还要包括 FDDI 的内容，仅仅为讲述历史，以防您碰巧发现它还在网络的某个旮旯里使用。

为了有良好的性能，FDDI 需要的 MTU 值要比为以太网调整的默认值高得多。4352（用 **ifconfig** 设置）就是一个合适的 MTU 值。关于 MTU 的更多信息请参见 12.3 节。

FDDI 标准规定了一种使用光纤传输介质，令牌传送速度为 100Mbit/s 的双环局域网，如图 14.4 所示。双环结构提供了用于数据传输的一个主环和用作备份以防环路中断（物理上或电气上）的一个次环。

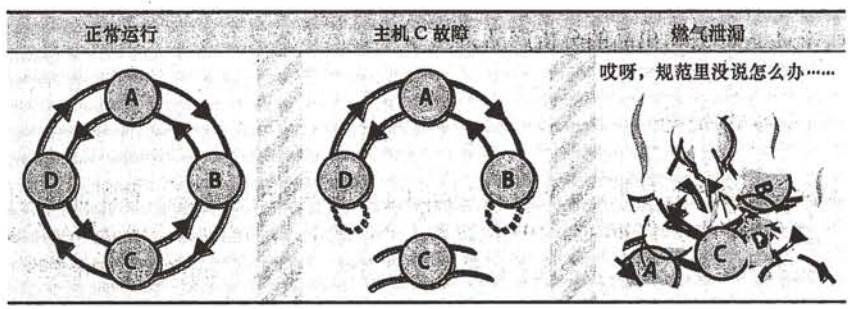


图 14.4 FDDI 双令牌环

可以把主机连接到两个环上（这就是 A 类或“双接”主机），也可以只把主机连接到主环上（这就是 B 类或“单接”主机）。大多数情况下，主干路由器和集中器都是双接的，而工作站是单接的，通常要通过“集中器”（也就是一种光纤集线器）连接主机。

[7] FDDI 也被接受为 ISO 的一个标准。

www.TopSage.com

令牌环系统的一个优点是：对网络的访问是由一种确定性协议来控制的。因为没有冲突，所以在高负荷下，网络性能也不会像在以太网上那样降低。在为多台客户机提供服务时，许多令牌环系统都能够以它们额定容量的 90%~95%运行。

对于物理介质，FDDI 标准建议两类光纤：单模光纤和多模光纤。“模”实质上是以一定的角度进入光纤的光线束。单模光纤只允许一种频率的光通过，这样就需要激光作为发射光源[8]。多模光纤允许多束光线通过，通常是用更便宜而且更安全的 LED（发光二极管）来驱动的。与多模光纤相比，单模光纤的距离更长。实际上，对 FDDI 而言，最常用的是 62.5μm 的多模光纤。

FDDI 使用的光纤连接器有几种标准，而且这些标准随厂商而异。不管使用什么连接器，一定要记住：干净清洁的光纤连接是可靠运行的关键。虽然有用户自己用的光纤终结设备，但我们建议您让专业连线公司安装光纤段的端部。

14.5 ATM：有前途的（但被痛苦地击败的）LAN

ATM 代表异步传输模式（Asynchronous Transfer Mode），但一些人坚持认为它表示“又一种技术错误（Another Technical Mistake）”。一位数据通信产业的发言人把它描述成“电信公司企图把您的网络问题转变成他们知道怎样收费的东西的尝试。”

ATM 在技术上是很特别的，因为它提出了一种思想，即用小的、固定长度的包（称为“信元”）是实现千兆比特网络最有效的方法。ATM 还承诺了一些其他介质传统上没有承诺的性能，包括带宽预留和服务质量保证。

在 ATM 53 字节的信元之上，为信元传输描述了 5 个 ATM 适配层（Adaptation Layer，AAL）。表 14.4 总结了每个适配层的目的。

我们不清楚在现实中该如何使用 AAL 2。目前也没有规定它的标准。AAL3 和 AAL4 显得非常相似，因此二者结合在一起。必须实现 ATM 的一群厂商很不喜欢 AAL3 和 AAL4，因为它们的开销太大了。这些厂商开发出了自己的解决方案 SEAL（Simple and Efficient Adaptation Layer，简单和高效适配层），很快就变成了 AAL 5。

在 20 世纪 90 年代，ATM 作为一种“全能（all-in-one）”的交换式网络介质，在针对 LAN、WAN 和 MAN 需求的市场里进行了全面推广。现在，ATM 几乎死掉了，它只保留在了 WAN 的环境中，大型电信公司仍然在这个环境中试着利用他们在 ATM 硬件上的错误投资。

ATM 交换机厂商继续努力地推销着自己的产品，在许多地方都可能订购到 ATM 电路。不过，在新的实施项目上考虑采用其他技术而不是 ATM，可能才是个好主意。

表 14.4　　ATM 适配层

AAL	应　用
1	位速率恒定的应用，比如语音（要求延迟受限）
2	位速率可变的应用，要求延迟受限
3	面向连接的数据应用
4	无连接的数据应用
5	一般数据传输（特别是 IP 流量，取代了 3 和 4）

[8] 在光纤网络中使用的大多数激光都是 1 类（Class 1），这要么意味着“照进眼睛是安全的”，要么意味着“照进眼睛不安全，但是设备的设计已经能防止在正常使用期间照进眼睛”。遗憾的是，“正常使用”或许并不包括在断光纤上花时间，所以真的不能保证安全。所以不要把激光照进您的眼睛，即使所有摆酷的小伙子们都这么做（译者注，以此判断光纤是否通）。

www.TopSage.com

14.6 帧中继：被牺牲掉的WAN

帧中继是一种WAN的技术，它提供包交换数据服务，通常其成本也比较合理。尽管人们声称它没有100%的准确性，但帧中继常被说成是重入市场的X.25，X.25是一种20世纪70年代中期出现的差劲的包交换技术。幸好它的应用相当广泛，支持它的设备、软件和人员已经发展到很强大的程度，而且表现不错。

传统上，希望连接到远程站点的用户要从电信公司购买一条专用电路，比如56 kbit/s DDS线路或T1线路。这些点对点的数据电路每天24小时保持连接。遗憾的是，这类连接通常相当昂贵，因为它要求电信公司提供链路的专用设备和带宽。

相反，帧中继采取的是一种“规模经济”模式。电信公司建立一个连接其若干交换局的网络（通常也叫做“云[cloud]”[9]）。（译者注：central office。世界上第一个电话交换局成立于美国康涅狄克州的New Haven城，当时仅有21个用户，由于交换局位于用户所在地的中心，所以称之为central，用户在打电话时对接线员的称呼为“Central!”。时至今日，现代美国英语中已用“Central Office”取代“Central”，用“Operator!”取代“Central!”。在CCITT的术语表中虽然不收录Central Office这一美国名词，但在该文件附录中的多国文字词汇表中，与英语exchange相对应的法语名词和西班牙语名词均为central。所以central office实际上是交换局而不是总局。）用户以小包的形式为远程站点提供数据。电信公司通过合适的交换局来交换包，最终把它们传递到目的地。在这种模型中，您和电信公司在任何一秒的时间里都在赌博，赌通信的总量不会超过网络带宽（超过的情形委婉地称为“超额使用”）。

路由器在帧中继连接上封装IP流量。包在不可见的PVC（永久虚拟电路）上进行交换，这使得包只通过您为之付费的站点。这些PVC提供了一定程度的私密性保护，与连接到这个帧中继网络的其他站点分开。

帧中继的最大优点是它一般比较便宜。但在“一分钱一分货”的世界里，您会发现帧中继的性能并不好。帧中继连接有一些包交换的开销，而且在繁忙的情况下，链路速度可能会降低。

14.7 ISDN：本土上的WAN

综合业务数字网（Integrated Services Digital Network，ISDN）是电信公司提供的一种服务，它的形式多样。最常见和可用的形式为基本速率接口（Basic Rate Interface，BRI）ISDN，它实际上是一根数字电话线，这条线提供两个拨号64kbit/s“B”信道和一个16kbit/s信令“D”信道。每个B信道既能够用于语音也能够用于数据（在一个64kbit/s通道上可以承载一条语音线路）。

ISDN以合理的成本（每月30~150美元，取决于您住在哪里）提供了相当高速的数字线路。称为终端适配器的设备把电话线转换成RS-232这样的计算机熟悉的接口。它们的使用方法（和定价）非常像调制解调器。大多数适配器能够集成两个B信道，得到一个128kbit/s的数据通道。

我们能够用ISDN来代替正常的拨号连网技术，也可以把它作为一种广域技术，用一台路由器或网桥经这条线路连接到远程站点。

虽然许多美国的电信公司安装了兼容ISDN的交换机，但他们还是没有确定应当如何把ISDN投入市场和提供支持[10]。只有在几个地区，您才可以打电话给电信公司订购ISDN线路。一些提示：要

[9] 这个名字并不贴切，因为在帧中继网络中，从来都搞不清楚天气预报到底指什么。风暴？下雨？下雪？还是冰雹？

[10] 因此有人认为ISDN表示“它还不做事（It Still Does Nothing）”。

www.TopSage.com

确保您是与电信公司的业务部门打交道，因为 ISDN 通常分属这个部门。在许多地区，尽管实际上可以得到 ISDN 服务，但要遇到一个真正听说过它的人还真要费上一番周折。

14.8 DSL 和 CM：大众的 WAN

在商业公司和其他大型数据设施之间传送大量数据是很容易的。为了把比特位从一个位置传送到另一个位置，像 T1、T3、SONET、ATM 和帧中继这样，由通信公司提供的技术提供了相当简单的渠道。然而，为了连接个别住户和家庭办公室，这些技术并不是现实的选择。它们的成本太高，而且它们所需要的基础设施通常也不容易获得。

数字用户线路（Digital Subscriber Line，DSL）使用普通的铜制电话线以高达 7Mbit/s 的速度（虽然典型的 DSL 连接速度是在 256kbit/s 和 768kbit/s 之间）传输数据。因为大多数家庭已经有电话线，所以使用 DSL 来提供从电话公司到家庭之间“最后一英里”的连接才是一种可行的途径。DSL 连接通常到一个机盒为止，这个机盒充当了一台 TCP/IP 路由器，并且提供了到家庭内其他设备上的一条以太网连接。一般情况下，DSL 比 ISDN 便宜，而且更快，所以现在对于家庭用户而言，DSL 是优先选择的技术。

DSL 与一般的 POTS（Plain Old Telephone Service，简易老式电话业务）以及 ISDN 连接不同，后两者需要您向端点拨号，而 DSL 是始终连接着的专用网络服务。这个特性使 DSL 更具有吸引力，因为当一个用户想传输数据时，不必因为设置或连接而有延迟。

DSL 有几种形式，因此经常叫做 xDSL，x 代表一种特定的技术分支，例如：A 代表非对称（asymmetric）、S 代表对称（symmetric）、H 代表高速（high speed）、RA 代表速率自适应（rate adaptive），而 I 代表 DSL-over-ISDN（对于距离交换局太远，不能支持速度更快的 DSL 技术的地方很有用）。在您所在的地区能得到的确切技术种类和数据传输速率取决于电信公司或通信公司选择安装的交换局端设备。

为连接亿万个家庭的“最后一英里”而展开的竞争现在已经白热化了。这方面的竞争受到了政治关注、吸引了大量资本，而且被过分炒作。DSL 方案利用了铜线的基础设施，这在 ILEC（Incumbent Local Exchange Carriers，本地电信运营商）之间是很常见的。经历了 20 世纪 80 年代到 90 年代网络的演变，ILEC 想在基础设施投资上获得更高的利润幅度。

有线电视公司在多数邻近区域已经有了光纤基础设施，他们正在推行自己的“最后一英里”解决方案，这种方案提供了类似的高带宽（虽然是非对称的）到户连接。电缆调制解调器（cable modem，CM）行业最近已经受到数据标准的启发，正在推出有线服务数据接口规范（Data Over Cable Service Interface Specification，DOCSIS）标准。这个标准为电缆调制解调器和有线电视公司使用的设备定义了技术规范，它能够让各种品牌的设备彼此间进行互操作。

总之，我们可以把电缆调制解调器和 DSL 技术之争归结为“市场营销预算的高低”之争。DSL 有其固有的优势，因为每条连接都是到特殊客户的私有线路，邻里间的电缆调制解调器用户却要共享带宽，而且可能会窃听对方的流量。

14.9 网络的前景

仔细地看一看上面介绍的各种技术，就会发现一个共同点：简单廉价的技术正在取得成功，而复杂昂贵的技术很快就会被淘汰。这种情况会把我们带到什么样的发展道路上去呢？

以太网因为出奇的廉价而已经彻底击败了它的竞争对手。如今，实现以太网相当简便，您甚至可以买到带有以太网接口的微波炉。以太网的规模扩大得很快：在许多机构中，20 世纪 80 年代早期启

www.TopSage.com

用的 10Mbit/s 以太网的基础设施仍在实际使用，并且连接到了 100Mbit/s 和 1Gbit/s 的网段上。现在已经有了 10Gbit/s 光纤以太网，我们很快就会看到它在铜导线上得到广泛使用。我们期望看到这种趋势能够继续下去，用越来越快的交换硬件来连接一切。

在“连接到户”的战线上，DSL 为旧的铜线设备带来了新的活力。电缆调制解调器应用的增长已经带给数百万家庭高速的接入（还有实实在在的安全问题）。

在所有这些新发展中最好的一点是：不管介质是什么，也不管它的速度有多大，TCP/IP 都能够与之兼容。

14.10 网络测试和调试

向以太网（和其他以 UTP 为基础的技术）大规模迁移的一个主要的优点是网络调试方便。因为可以逐条链路地分析这些网络，所以经常能够在几秒而不是几天内隔离出硬件问题。

调试网络的关键在于要把网络分解成许多组成部分，并对每个部分进行测试，直到把损坏了的设备或电缆隔离开来为止。通常交换机和集线器上的“闪烁灯”（例如“链路状态”和“包流量”指示灯）可以提供故障源的直接线索。要让那些指示灯按您的要求起作用，为整个网络的连线方案建立第一流的文档至关重要。

对大多数任务而言，有合适的工具是能够正确完成工作而且不拖延时间的一个重要因素。市场上提供了两种主要类型的网络调试工具（虽然它们正在很快地合并到一起）。

第一类是手持式电缆分析仪。这种设备能够测量一段给定电缆的电气特性，包括它的长度（采用了称为“时域反射计”的时髦技术）。通常，这些分析仪也能够指出简单的错误，例如断线或错误接线的电缆。在 LAN 电缆分析上，我们最喜欢的产品是 Fluke LanMeter。这是一种全能的分析仪，它甚至能够在网络上运行 IP 的 ping 命令。其高端版本带有自己的 Web 服务器，能够显示历史统计信息。对于 WAN（电信级）电路，T-BERD 线路分析仪的性能令人敬佩。T-BERD 和它的高端 LAN 测试产品 FIREBERD 系列都是由 Acterna 生产的。

第二类调试工具是网络嗅探器。这种设备分解网络上的包来寻找协议错误、配置错误和一般的混乱问题。虽然有商业的网络嗅探器，但是我们发现在笔记本上运行 Wireshark（www.wireshark.org）公司提供的免费程序通常是最好的选择[11]。

14.11 建筑物布线

不论您运行的是千兆以太网，还是只使用串行电缆，我们都建议您尽可能使用质量最好的导线，这会大大增加您在 10 年后仍能使用这类导线的可能性。一次对整个建筑物布线要比一次只连接一条线要便宜得多。

14.11.1 UTP 电缆的选择

当今市场上，5E 类导线的性价比是最好的。其正规结构是一个护套内有 4 对导线，这种结构对于从 RS-232 到千兆以太网的各种数据连接而言都适用。

5E 类规范要求在距离压紧连接处的半英寸之内还保持绞扭状态。这意味着每个护套内超过 4 对导线的任何缆线都必须系紧以保持绞扭状态，因为它提供了多条连接。

除了 5E 类缆线外，还必须使用 5E 类终结部件。我们有幸使用了 Siemon 公司制造的部件。

[11] 和许多流行的程序一样，Wireshark 经常容易受到黑客的攻击。要保证与目前的版本保持同步。

www.TopSage.com

大家网 TopSage.com

14.11.2 到办公室的连接

每个办公室只有一条连接显然是不够的。但应该使用 2 条还是 4 条？我们建议您使用 4 条连接，原因有以下几条：

- 它们可以用于串行连接（调制解调器、打印机等）；
- 它们可以与语音电话一起使用；
- 可以使用它们来满足外来设备或者演示设备的需要；
- 材料成本一般只有总成本的 5%~10%；
- 将最好情况的预测数翻倍通常是比较好的估计值；
- 一次完成比以后再加线要便宜很多。

如果您正在对整个建筑物布线，可能会考虑在走廊、会议室、餐厅和浴室安装几个插座，当然还要在天花板上装无线的 AP。但是不要忘记保持安全，把公开访问的端口放在一个“客用”VLAN 上，这个 VLAN 无法访问您的内部网络资源。

14.11.3 布线标准

现代建筑物为了支持内部发生的各种活动，通常需要庞大而复杂的布线基础设施。走进电信公司的普通机房，您将惊奇地发现，墙上布满了颜色相同、没有任何标记的导线。

为了增加缆线的可追踪性，并使建筑物的布线标准化，TIA（Telecommunications Industry Association，电信行业协会）在 1993 年 2 月发布了商业建筑物电信基础设施的 EIA-606 管理标准。EIA-606 规定了对电信基础设施的标识和建档的要求和指导。

EIA-606 包括的条款有：

- 终结点硬件；
- 电缆；
- 电缆走线；
- 设备空间；
- 基础设施颜色编码；
- 标准部件的符号。

它特别规定了布线使用的标准颜色。表 14.5 给出了详细规定。

表 14.5 EIA-606 颜色规范

终端类型	颜色	编码[a]	说明
定界点	橙色	150C	交换局终结点
网络连接	绿色	353C	也用于辅助电路终结点
常见设备[b]	紫色	264C	主要的交换/数据设备的终结点
一级主干	白色	–	电缆终结点
二级主干	灰色	422C	电缆终结点
站	蓝色	291C	水平电缆终结点
建筑物间主干	棕色	465C	校园电缆终结点
杂项	黄色	101C	维护、警报等
集团电话系统	红色	184C	–

a. 按照 PMS（Pantone Matching System，潘通配色系统）。
b. PBX（private branch exchange，用户交换机）、主机、LAN 和多路复用器等。

www.TopSage.com

Pantone 公司现在出售专门的软件，在纸用油墨、纺织品染料和彩色塑料的各个 Pantone 系统之间建立映射关系。您甚至可以给线路连线、安装人员的制服以及布线文档协调上色。再想一想……

14.12 网络设计问题

本节讲述网络的逻辑和物理设计。它的目标是安装中等规模的网络。本节介绍的设计思想可以扩展到几百台主机上，但对于仅有 3 台机器的网络显然是小题大做，而对数千台机器的网络又显得力不从心。我们还假定：您有一笔充足的预算，而且正在从零开始做起，这种假定可能只是部分正确。

大多数网络设计包含以下方面的规范：

- 要使用的介质类型；
- 电缆的拓扑结构和路由；
- 交换机和路由器的使用。

网络设计的另一个关键问题是拥塞控制。例如，NFS 和 CIFS 对网络资源的要求相当高，因此不要在主干电缆上提供文件服务。

下面各小节讲述的问题是在任何一个网络设计中都必须考虑的典型问题。

14.12.1 网络结构与建筑物结构

网络结构通常比建筑物结构更为灵活，但这两种结构必须共存。如果幸运的话，您能够在建造建筑物之前确定网络的结构，那简直是太好了。对于我们大部分人而言，情况则是建筑物和物业管理部门都已经存在，并且还有点儿强硬。

在现有的建筑物中，网络必须利用建筑物的结构，而不是“违抗”这种结构。除了高压电线以及水或煤气管道外，现代建筑物通常还有数据和电话线的管道。它们经常用吊顶布线，这对网络安装者来说很方便。许多园区和机构使用便于网络安装的地下管道。

必须保证防火墙的完整性[12]。如果沿着防火墙内走线，洞内必须保暖，而且要在洞内填充防火材料。在选择电缆时，要考虑通风系统回风区的要求。如果被抓住违反了防火条例，可能会受到惩罚，而且要纠正所造成的问题，即便意味着要拆掉整个网络并正确地重建网络，也仍然会强制执行。

网络的逻辑设计必须适应它所服务的建筑物的物理限制。在规划网络时，请记住，绘制逻辑上良好的一个解决方案很容易，随后却会发现它在物理上很难或不可能实现。

14.12.2 现成的网络

计算机网络是本节讨论的中心，许多单位已经有了能够传输数据的有线电视网络和电话网络。通常它们包含了光纤链路。如果您的单位准备安装一个新的电话系统，那么可以多买些光纤，和这个系统一起安装起来。

几年前我们有过这样的机会，而且询问过承包商他们是否可以为我们安装一些光纤。他们说“当然可以，免费安装”，当我们拿出一车光纤让他们安装时，他们有些生气。

14.12.3 扩展

人们很难预测十年后的需要，在计算机和连网领域尤其如此。因此设计网络时考虑到扩展和带宽

[12] 这类防火墙是混凝土、砖或者防火材料盖的墙，可以阻止火焰在建筑物内的传播和燃烧。尽管它与网络安全防火墙有很多的不同之处，但它可能也同样重要。

www.TopSage.com

的增加是很重要的。在安装电缆时，特别是在偏远、难以到达的地方安装网络时，应当将电缆长度保持为实际需要长度的3~4倍。记住：安装成本主要是人工，而不是材料。

即使您目前不准备使用光纤，在给建筑物布线时安装一些光纤也是很明智的，尤其是在以后很难安装电缆的情况下。既要安装多模光纤，也要安装单模光纤，这是因为将来所需要的光纤类型常是您没有安装的类型。

14.12.4 拥塞

网络就像一条链子一样，其好坏是由其最薄弱、最慢的环节决定的。与其他许多网络结构一样的是，随着网络负载的增加，以太网的性能会下降。

过载的交换机、不匹配的接口和低速链路都可能导致拥塞。通过创建子网和使用像路由器这样的互连设备把本地流量隔离开来是很有帮助的。我们也能够使用子网来隔离那些实验用的机器，如果没有简单的方法把那些实验机器从物理和逻辑上与网络的其他部分隔开，那么只包括这几台机器的实验就很难进行。

14.12.5 维护和建档

我们已经发现，网络的可维护性与其建档质量有密切的关系。准确、完备、保持最新的文档必不可少。

应该在所有终结点给电缆做上标记，沿缆身每隔几英尺也应当做上标记，这样一来，在天花板或墙上发现电缆时，就能够很容易地识别出它们[13]。最好在通信室里留一份本地电缆图的副本，这样一旦发生变化，可以在现场立即更新它们。每隔几个星期，应该有人把变化复制下来并输入到布线数据库中。

在各个主要的传送中心之间，以交换机或者路由器的形式连接起来的结合处会有助于调试，因为可以使网络各部分隔离开来单独调试。在不同行政区域和管理域之间设立结合部也有利于工作。

14.13 管理问题

如果要让网络正常运行，有些事情要集中做，有些要分散做，而有些要在本地做。人们需要制订合理的基础规则和“好市民”准则，并就此达成共识。

典型的网络环境包括：

- 建筑物之间的主干网络；
- 连到主干网络的部门子网；
- 一个部门内的组子网；
- 与外部世界的连接（例如 Internet 或现场办公室）。

网络设计和实施的几个方面必须包括，全站范围的控制、负责、维护和财政支持。因为各部门都在设法将本部门的成本减到最小，所以，以每条连接扣款方式付费的网络以非常奇特但又可预测的方式成长。集中控制的基本目标是：

- 网络设计，包括子网、路由器、交换机等的使用；
- 主干电缆自身，包括和它的连接；

[13] 有些电缆生产商会在电缆上每隔几英尺预先做上标记。

www.TopSage.com

- 主机的 IP 地址、主机名和子域名；
- 协议，主要是要确保它们能够互操作；
- 到 Internet 的路由策略。

就某种意义来说，域名、IP 地址和网络名已经由像 ARIN 和 ICANN 这样的权威机构集中控制。不过，它们在网点内部的使用也应在本地做好协调。

中心权威部门会对网络的设计、容量和预期增长等有宏观的了解。该部门可以花钱购买自己的监视设备（以及雇佣运行它的人员）并且保持主干网络的健康。它能够坚持正确的网络设计，甚至这样做意味着要告诉某个部门购买一个路由器并建立一个子网来把这个子网连接到园区的主干网络上。这样的决定可能是必要的，这样一来，新的连接就不至于对现有网络造成负面影响。

如果一个网络能够为许多类型的机器、操作系统和协议提供服务，那么几乎有必要添置一台非常智能的路由器（例如 Cisco）来充当网络间的网关。

14.14 推荐的厂商

在过去超过 15 年的时间里，我们在世界各地做过网络安装工作，我们安装的产品不只一次地出过问题，它们不能够满足规格要求，或者并没有厂商宣传的那样好，价格过高，或者不能满足用户的一些期望。以下列出的是美国的一些厂商，他们的产品在现今仍然值得信任、推荐和使用。

14.14.1 电缆和连接器

AMP（现在属于 Tyco）
www.amp.com

Black Box Corporation
www.blackbox.com

Anixter
www.anixter.com

Newark Electronics
www.newark.com

Belden Cable
www.belden.com

The Siemon Company
www.siemon.com

14.14.2 测试仪器

Fluke
www.fluke.com

Acterna
www.acterna.com

Siemon
www.siemon.com

14.14.3 路由器/交换机

Cisco Systems
www.cisco.com

www.TopSage.com

14.15 推荐读物

BARNETT, DAVID、GROTH, DAVID 和 JIM MCBEE. *Cabling: The Complete Guide to Network Wiring(3rd Edition)*. Sybex. 2004.

SEIFERT, RICH. *Gigabit Ethernet: Technology and Applications for High Speed LANs.* Reading, MA: Addison-Wesley. 1998.

ANSI/TIA/EIA-568-A, *Commercial Building Telecommunications Cabling Standard* 和 ANSI/TIA/EIA-606, *Administration Standard for the Telecommunications Infrastructure of Commercial Buildings*，它们是电信业的建筑物布线标准。遗憾的是，它们不是免费的，请参考站点 www.tiaonline.org。

SPURGEON, CHARLES. "Guide to Ethernet." www.ethermanage.com/ethernet

14.16 习题

E14.1 如今，大多数写字楼都有计算机网络，而且也用 UTP 以太网布了线。需要集线器和交换机的某种组合来支持这些网络。在许多情况下，这两类设备能够互换使用。列出这两种设备各自的优缺点。

★E14.2 画一张简单的虚构网络图，把在您机房里的一台机器连接到 Amazon.com。其中要包括 LAN、MAN 和 WAN。给出每一组成部分所要用到的技术。图中要有集线器、交换机和路由器。

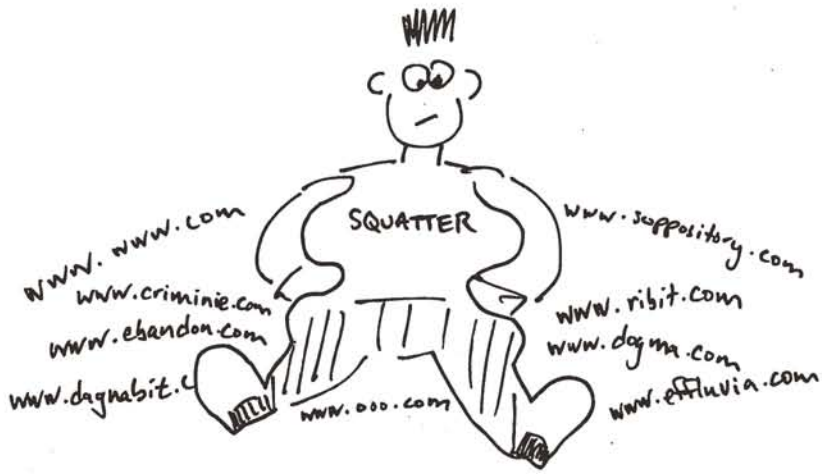
★E14.3 研究 WPA 的 TKIP 协议（Temporal Key Integrity Protocol，临时密钥完整性协议）。详述它比 WEP 有哪些优势，能防止哪些类型的攻击。

★★E14.4 TTCP 是一种测量 TCP 和 UDP 性能的工具（在 www.rpmfind.net 上找到它）。在两台连网机器上安装 TTCP，测量这两台机器之间的链路性能。如果您把缓冲区调大或者调小，带宽会发生什么样的变化？您所观测到的数字同物理介质的理论容量相比较如何？

www.TopSage.com

大家网
TopSage.com

第 15 章　域名系统



连接到 Internet 上的主机不计其数。它们属于那么多不同的国家、网络和行政机构，如何追踪它们呢？Internet 基础设施的两个关键组成部分将所有一切连为一体：域名系统（DNS）和 Internet 路由系统。前者跟踪主机是谁，后者跟踪主机如何连接。

本章（有人会说是一本迷你书）的内容与 DNS 有关。尽管 DNS 已经具有几种不同的用途，但它的基本任务还是建立主机名和 IP 地址之间的映射关系。用户和用户级的程序更愿意用名字来称呼机器，但下层的网络软件只能理解数字。DNS 提供了让它们保持皆大欢喜的粘合剂。DNS 还在电子邮件、Web 服务器访问以及许多其他服务的路由选择中扮演了重要角色。

DNS 是一个分布式数据库。“分布式”意味着我的网点保存着有关我的计算机的数据，您的网点保存着有关您的计算机的数据，而当某个网点需要查看另一个网点的数据时，我们大家的网点就以某种方式自动进行合作并共享数据。

我们介绍 DNS 的内容可以分为 3 个主要部分：

- DNS 系统概述：DNS 的历史、数据和协议；
- BIND：DNS 系统的一种专门实现；
- BIND 服务器的运行和维护，包括诸如安全这样的专题内容。

如果您需要建立或者维护自己站点的 DNS 服务器，而且已经对 DNS 的工作原理有一般性的概念，那么就可以放心继续阅读。

在开始介绍 DNS 的一般背景知识之前，先让我们简要回答每个人最爱问的一个问题：怎样把一台新机器加到已经使用 BIND 的网络上？下面是一份只讲步骤不解释或者定义任何术语的说明，它可能并不完全适用于您的本地系统管理策略和规范。谨慎使用吧。

www.TopSage.com

15.1 DNS 速成：添加新机器

如果您的网络设置了使用 DHCP（Dynamic Host Configuration Protocol，动态主机配置协议），那么可能不需要为 DNS 手工做任何配置。新机器连上网络之后，DHCP 服务器就会告诉它应该查询的 DNS 服务器。外界要用到的“主机到 IP 地址”的映射极有可能是在配置 DHCP 服务器的时候就设置好了，并且是通过 DNS 的动态更新机制输入的。

对于没有使用 DHCP 的网络来说，下面的步骤说明了如何通过复制和修改一台类似主机的现有记录来更新 DNS 的配置。

步骤 1：联系本地系统管理员和上游 ISP（Internet service provider，Internet 服务提供商），为新机器选择一个主机名和一个 IP 地址。

步骤 2：在同一子网上找到一台类似的机器。您将以该主机的记录作为新机器的模板。在本例中，我们使用一台叫做 templatehost.my.domain 的主机为模板。

步骤 3：登录到主名字服务器。

步骤 4：浏览名字服务器的配置文件，通常是**/etc/named.conf**：

- 从 options 语句中找到 directory 行，该行指明区数据（zone data）文件在您的网点上保存的位置（参考 15.11 节）。区文件（zone file）包含主机和 IP 地址的实际数据。
- 从 zone 语句中找到您的新 IP 地址所在网络的正向区文件（forward zone file）和反向区文件（reverse zone file）的文件名（参考 15.11 节）。

```
zone "my.domain"{
     type master;
...
```

步骤 5：进入区文件的目录并编辑正向区文件。找到先前认定的模板主机的记录。其内容看起来如下：

```
templatehost     IN  A   128.138.243.100
                 IN  MX  10  mail-hub
                 IN  MX  20  templatehost
```

您的文件版本可能不包括有 MX 的配置行，这个 MX 记录用于路由邮件。

步骤 6：复制这些记录并为新主机做相应的更改。这个区文件可能会根据主机名进行排序，请遵循现有的规则。还要更改该文件开头处 SOA 记录中的序列号（SOA 记录里 5 个号码中的第一个）。这个序列号只能增加。如果您的站点使用了一个任意选定的序列号，请加 1；或者，如果您的站点用日期作为制定序列号的规则，那么请将该字段设置为当前日期。

步骤 7：编辑反向区文件[1]，复制模板主机的记录并更新它。它看起来应该像这样：

```
100             IN PTR    templatehost.my.domain.
```

还必须更新反向区文件中 SOA 记录的序列号。

如果反向区文件显示的每个主机 IP 地址不只是它的最后一个字节，则必须反序输入这些字节。例如，某记录：

```
100.243         IN PTR    templatehost.my.domain.
```

对应于 IP 地址 128.138.243.100(这里的反向区和 138.128.in-addr.arpa 有关，而不是和 243.138.128.in-addr.arpa 有关)。

[1] 反向区可能保存在其他位置（例如在您的 ISP 的站点）。如果这样，则必须输入反向配置项。

www.TopSage.com

步骤 8：仍然登录到主名字服务器，运行 **ndc reload**，如果它是一台繁忙的服务器，那么只要重新载入您所修改的那些域（或者视图）即可：

```
# rndc reload forward-zone-name
# rndc reload reverse-zone-name
```

步骤 9：用 **dig** 检查配置，参考 15.16 节。即使还没有设置好这台新主机，也可以尝试 **ping** 或 **traceroute** 到新主机名。“host unknown（未知主机）”的消息说明您操作错误；“host not responding（主机无响应）”说明一切可能正常。

最常见的错误有：

- 忘记更新序列号就重启了名字服务器；
- 忘记在 PTR 反向配置项中的主机名末尾加一个圆点（.）。

15.2 DNS 的历史

Paul Mockapetris 在 RFC 882 和 883（1983 年）中正式规定了 DNS，并在 RFC 1034 和 1035（1987 年）中做了更新。DNS 包括两个关键的概念：层次结构的主机名和分散负责。

15.2.1 BIND 的实现

最早的 UNIX 实现是在 1984 年由 Berkeley 的 4 个研究生（Douglas Terry、Mark Painter、David Riggle 和 Songnian Zhou）完成的。在 20 世纪 80 年代中期，Kevin Dunlap 把它加入到了 Berkeley 的 UNIX 发布中，称为 BIND（Berkeley Internet Name Domain，伯克利 Internet 名字域）系统。Paul Vixie 和 ISC（www.isc.org，在 2004 年以前叫做 Internet Software Consortium，Internet 软件联盟）目前负责维护 BIND。在 2000 和 2001 年，ISC 利用从几家厂商、政府部门和其他机构获得的资金开发了一个全新的 BIND 版本——BIND 9。

ISC 还提供对这些产品的多种类型的支持，包括配置帮助、BIND 和 DNS 的课程、甚至定制编程。对于使用开放源代码软件之前必须拥有支持合同的网点来说，这些服务确实是一种实惠。几家公司用服务合同作为一种对 ISC 的捐助方式——他们购买了昂贵的合同，却从来没有请求过帮助。

多亏 Nortel 做的移植工作，Windows 和 UNIX/Linux 一样能用上 BIND 了。因为 DNS 协议是标准，所以 UNIX 和非 UNIX 的 DNS 实现都能互操作和共享数据。许多站点运行 UNIX 服务器向它们的 Windows 台式机提供 DNS 服务，这种组合效果颇佳。

RFC 1034 和 1035 仍被认为是 DNS 的基准规范，但是在过去的 10 年中，已相继产生了超过 40 个其他的 RFC，它们在协议和数据记录的多个方面进行了精心设计（请参考本章结尾的列表）。目前，没有单独的标准或 RFC 能将所有内容总结到一起。在历史上，DNS 已经或多或少被定义为“BIND 所实现的功能”，尽管随着其他 DNS 服务器的出现，这样下定义变得越来越不够准确了。

15.2.2 其他的 DNS 实现

一开始，BIND 是得到广泛使用的惟一一种 DNS 实现。现如今又有了好几种别的实现，它们中间有开放源代码的，也有商业产品。许多 DNS 实现都没有实现在标准化过程中出现的那么多 DNS RFC 所制定的全部规范。表 15.1 列出了比较流行的 DNS 实现以及到哪里可以获得更多相关资料。

www.TopSage.com

表 15.1 一些流行的 DNS 实现

名 称	作 者	来 源	说 明
BIND	ISC	isc.org	权威的或者缓存的服务器
NSD	NLnet Labs	www.nlnetlabs.nl	仅限于权威的服务器
PowerDNS	PowerDNS BV	www.powerdns.com	仅限于权威的服务器
djbdns[a]	Dan Bernstein	tinydns.org	违反了一些 RFC 的规定
微软的 DNS	微软	microsoft.com	被指责有大量毛病
ANS,CNS	Nominum	www.nominum.com	权威的或者缓存的服务器

a. 也称为 tinydns，它是 djbdns 软件包的服务器组件。

ISC 持续对域名的调查跟踪着各种 DNS 实现以及使用每种实现的域名服务器的数量。到 ISC 的网站 www.isc.org，沿着 ISC Internet Domain Survey（ISC Internet 域名调查）、Latest Survey Results（最新调查结果）和 Domain Server Software Distribution（域名服务器软件的分布）点下去，就可以看到当前的统计数据。像 Infoblox（www.infoblox.com）这样的 DNS 设备有许多大型站点在用，但在这份报告中还没有出现。

我们在本书里只讨论 BIND，它被认为是 DNS 的参考实现。它也是迄今为止使用最广泛的实现，对于大多数站点来说都是一种合适的选择。BIND 遵守了 IETF 的所有标准和推荐标准，经常在 IETF 的规范还没有完成之前就已经实现了相应的功能。这样是有好处的，因为有些符合标准的功能却可能有缺陷——把它们包括到 BIND 里来会让问题被确认出来，并在写入“法律”之前改正。

NSD（name server daemon，名字服务器守护进程）由阿姆斯特丹的 NLnet Labs 于 2003 年开发。它提供了一种快速、安全和权威的名字服务器，适合于根和顶级域名服务器使用（一台权威服务器适合于回答对您的名字域内主机的查询，但是它不能回答用户对其他域的查询）。

PowerDNS 是一种开放源代码的权威名字服务器，它提供了一种具有独一无二灵活性的后端系统。DNS 数据能从文件或者其他资源得来，这些资源构成了一份的长长清单：MySQL、Oracle（8i 和 9i）、IBM 的 DB2、PostgreSQL、微软的 SQL Server、LDAP、ODBC、XDB、甚至一个 UNIX 管道。

djbdns 是一种可供选择的名字服务器软件包，它由一个叫做 **tinydns** 的权威服务器和一个叫做 **dnscache** 的缓存服务器组成。它声称自己很安全且速度非常快，尽管我们掌握的一些测量数据还不足以下结论。它的主要缺点是频繁而且有意地违反 DNS 的标准，使得它难以和其他 DNS 服务器互操作。

微软在 Windows 中提供了一个 DNS 服务器，但是微软的实现有它自己的特别之处和差异。它能够和 BIND 互操作，但也有可能用不必要或者错误格式的包扰乱网络。

Nominum 是为 ISC 编写 BIND 9 初始版本的承包公司，它也销售自己的名字服务器和网络管理工具。Nominum 的服务器速度非常快，而且包含了当前提出的大部分标准的功能。

15.3 谁需要 DNS

DNS 定义了：

- 用于主机和 IP 地址的一个有层次结构的名字空间；
- 作为一个分布式数据库实现的主机列表；
- 一个查询这个数据库的“解析器（resolver）”；
- 改进的路由电子邮件的机制；
- 一种用于在网络上查找服务的机制；
- 一种用于交换命名信息的协议。

www.TopSage.com

站点需要有 DNS 才能完全成为 Internet 的成员。光靠维护一个本地的/etc/hosts 文件，为用户可能希望与之联络的每台主机建立映射的做法是不可行的。

每个站点维护着构成全世界 DNS 系统的分布式数据库的一个或多个部分。您这部分数据库由两个或多个文本文件组成，这些文件包含有您的每台主机的记录。每条记录一行，由一个名字（通常是主机名）、一个记录类型和一些数据值组成。

例如，“正向”文件中的以下几行：

```
bark        IN   A    206.168.198.209
            IN   MX   10 mailserver.atrust.com.
```

和“反向”文件中的以下几行：

```
209                 IN   PTR  bark.atrust.com.
```

把“bark.atrust.com”同 IP 地址 206.168.198.209 联系起来，而且把目的地址是这台机器的电子邮件重新路由给主机 mailserver.atrust.com。

DNS 是一种客户机/服务器系统。服务器（“名字服务器”）将数据从 DNS 文件载入内存，并用它来答复内部客户机以及 Internet 上的外部客户机和其他服务器的查询。您的所有主机都应该是 DNS 客户机，但只有相对来说极少的主机需要作为 DNS 服务器。

如果您所在的单位比较小（就一个网络上的几台主机），您可以在某一台主机上运行服务器，或者要求 ISP 为您提供 DNS 服务。具有几个子网的中等规模的网点应该运行多个 DNS 服务器，以减少查询的时间延迟并提高可靠性。一个大型网点可以将 DNS 域划分为几个子域，在每个子域上运行若干台服务器。

15.4 DNS 名字空间

DNS 名字空间被组织成一个数学家称之为“树”的结构，每个域名对应这棵树的一个结点。DNS 名字树的一个分支将主机名映射到 IP 地址上，另一个分支把 IP 地址映射回主机名。前一个分支叫做“正向映射（forward mapping）”，与之相关联的 BIND 数据文件叫做“正向区文件（forward zone file）”。地址到主机名的分支是“反向映射（reverse mapping）”，它的数据文件叫做“反向区文件（reverse zone file）”。说来遗憾，许多站点没有它们的反向映射。

每个域代表名字空间中的特定一块，并由一个行政实体进行松散的管理。树的根称为“.”或者“点”，在其之下是顶级（或根级）域。

出于历史的原因，目前使用两类顶级域名。在美国，顶级域最初代表组织性或行政性结构，并且是 3 字母的名字，例如 com 和 edu。这其中的有些域（主要是 com、org 和 net）也在美国以外使用。它们称为普通顶级域，或缩写为 gTLD（generic top-level domains）。

过去，顶级域一直相当固定，但是 ICANN 在 2000 年底批准了 7 个新增的顶级域：biz、info、name、pro、museum、aero 和 coop[2]。再后来又在 gTLD 列表中加入“job”。这些域名现在都在运行而且可以使用。biz、info 和 name 域称为“不付费”gTLD，面向任何人开放；museum、aero、jobs、pro 和 coop 是“付费”TLD，限于特定类型的注册人。

表 15.2 列出了最重要的 gTLD 以及它们原本的用途。在 com 域中的好名字变得稀缺起来以后，注

[2] ICANN 代表 Internet Corporation for Assigned Names and Numbers（Internet 名称和编号分配组织），它是 Internet 的管理机构，请参考 12.1.2 节以获得有关 ICANN 的更多信息。

www.TopSage.com

大学网 TopSage.com

册机构就开始提供 org 和 net 的域名，而不再考虑这些域最初的限制。表 15.2 中左边一列里的域是最初的顶级域，它们可以追溯到大约 1988 年；右边一列包括了 2001 年新增加的域名。

表 15.2 普通顶级域

域	用 途	域	用 途
com	商业公司	aero	航空运输业
edu	美国教育机构	biz	商业
gov	美国政府机构	coop	企业
mil	美国军事机构	info	不限用途
net	网络提供商	jobs	人力资源
org	非盈利组织	museum	博物馆
int	国际性组织	name	个人
arpa	IP 地址树的定位点	pro	会计、律师等

美国以外的大多数域都使用两个字母的 ISO 国家代码。这些域叫做 ccTLD，即“country code top-level domains（国家代码顶级域）”。在同一个全球域名空间中，地理性和组织性的 TLD 并存。表 15.3 给出了一些常见的国家代码。

美国以外的一些国家使用二级域建立了组织性的层次结构。它们的命名规则各不相同。例如，在美国的一个学术机构的域名可能是 edu，而在日本则可能是 ac.jp。

表 15.3 常见的国家代码

代 码	国 家	代 码	国 家	代 码	国 家
au	澳大利亚	fi	芬兰	cn	中国
ca	加拿大	fr	法国	ch	瑞士
br	巴西	jp	日本	mx	墨西哥
de	德国	se	瑞典	hu	匈牙利

顶级域“us”有时也在美国使用，它主要用于地域性域名，例如 bvsd.k12.co.us，代表科罗拉多州的 Boulder Valley School District。“us”这个域名从来不和组织性域名结合起来使用——没有“edu.us”（尚未出现）。使用“us”域名的好处在于其注册免费或便宜，从 www.nic.us 那里可以获得更多的详细信息。对“us”下二级域名的限制（以前限制只能使用美国的州名）已经放开了，可能有类似 evi-nemeth.us 这样的域名出现。

有些情况下，惟利是图的域名贩卖商购买了一个国家的整个域名空间。例如，代表摩尔多瓦的域名“md”如今已经卖给美国马里兰（Maryland，MD）州的医生和居民。另一个例子是图瓦卢，其国家代码是“tv”。第一宗这样的销售案例是汤加（“to”），当前最活跃的是纽埃岛（“nu”），或许最引人注目的是来源于土库曼斯坦的“tm”。对于有受到欢迎的双字母代码的国家，这些交易有时公平，有时不公平。

域名抢注也十分普遍：人们注册他们认为将来会被申请注册的域名，然后把域名转卖给名字被他们抢注的公司。几年前，科罗拉多州所有滑雪场的域名被同一个人注册，当这些域名在网上具有知名度时，这个人把它们转卖给了各个滑雪场，从而赚了很大一笔钱。

com 域中好域名的价钱介于几千到几百万美元之间。我们的域名 admin.com 卖了 50 000 美元，我们是在几年前/Sys/Admin 杂志已经注册了 sysadmin.com 的时候才注册这个域名的。迄今为止（或者，至少是公开的最高记录）的最高价是在技术股大涨的时期，business.com 卖的 750 万美元。现在，2 万到 10 万美元是一个常见的价格范围，但仍然会出现数百万美元的买卖，2004 年 7 月 CreditCards.com

www.TopSage.com

卖了275万美元，就是一个例子。

Internet创业人Dan Parisi曾有希望将以前的黄色网站whitehouse.com买个数百万美元的价钱，但这个站点在2004年早些时候被封。从那以后，连续有几家不同的公司用过那个域名，但是准确的合同条款以及资金细节则一直没有公开。

目前，有效的域名只包括字母、数字和短划线。一个域名的每个组成部分不能超过63个字符，完整的域名全长不超过256个字符。DNS系统的国际化以及对非ASCII字符集的支持最终改变了所有的命名规则，但是眼下其他字符集的域名还是被映射回ASCII字符集了，参考15.6节。

域名不区分大小写。就DNS而言，“Colorado”、“colorado”，以及“COLORADO”都是一样的。目前的DNS实现在做比较时必须忽略大小写，但是在传送时则要保持大小写不变。过去的通常写法是顶级域名全大写，二级域名首字母大写。如今，人们厌倦了在键入上花费指力，全部小写成了通常的写法[3]。

Internet上一台主机的全名（fully qualified name）是由它的主机名加上域名构成的。例如，boulder.colorado.edu是位于Colorado大学（University of Colorado）的主机boulder的全名。其他网点可以使用主机名boulder而不会造成冲突，因为它们的全名不同。

在DNS系统中，全名是以一个点（.）来结束的，例如“boulder.colorado.edu.”。没有最后的那个点则表示一个相对地址。根据一个相对地址的使用环境不同，可能要添加不同的额外部分。对于DNS的日常用户来说，通常并不需要知道最后要有点的习惯。事实上，如果您擅自加了这个点，某些系统（例如电子邮件）还会中断工作。

一台主机拥有多个名称的情况很常见。如果我们希望让主机名反映出它所提供的服务，那么主机boulder.colorado.edu也可以是www.colorado.edu或ftp.colorado.edu。事实上，让诸如www这样的服务主机名具有“流动性”是个不错的做法，这样一来，您便可以将服务器从一台机器转移到另一台机器，而不需要更改任何机器的基本名称。用CNAME结构能够分配更多的名字，参考15.7.7节。

当分给我们域名colorado.edu时，也就保证我们的colorado在edu域内是惟一的。我们还要进一步按系将这个域划分为子域。例如，计算机科学系的主机anchor在Internet上称为“anchor.cs.colorado.edu”。

创建每一个新的子域必须与上级域的管理员取得协调以保证惟一性。父域配置文件中的项会把域名空间的权力授予子域。

15.4.1 域名管理

顶级域com、org、net和edu的管理以前是由Network Solutions公司与国家科学基金会（National Science Foundation）签定合同协调完成的。这种垄断的情况如今已经改变了，其他组织也得到允许可以在这些gTLD中注册域名。其他顶级域，例如各个国家的域名，都由区域性的组织来维护。

已经出现了让私营公司管理其自身的顶级域的多种提议，可能在不久的将来会出现更多的顶级域。参考www.icann.org以获得最新信息。

大多数ISP提供有偿的域名注册服务。他们代表您与顶级域的权威机构打交道，并配置他们的服务器以处理在您的域内的域名解析。依赖于ISP服务器的弊端在于您丧失了对自己域的管理事务的直接控制。

即使您希望管理自己的DNS服务，也必须与ISP进行协调。大多数ISP在其CIDR地址块内提供IP地址的反向DNS映射。如果您接管了自己地址的DNS管理，就要确保您的ISP禁止对那些地

[3] BIND保留大小写，但是有些实现（例如，微软和djbdns的DNS实现）按照它们自己的偏好而改变了大小写的规定。还是谨遵标准为好。

www.TopSage.com

址的 DNS 服务并把这一职责授权给您。请参考第 12.4 节以获得有关 CIDR 的更多信息。

只要可能，就应当在相同的位置管理一个域的正向映射和反向映射。一些 ISP 乐于让您管理正向映射文件，但是却不太情愿放弃对反向映射的控制。像这样分开管理可能导致同步方面的问题。请参考第 15.7.8 节，了解适用于极小范围地址空间授权的一种好窍门。

DNS 域应该（事实上是必须，请参考 RFC1219）由至少两台服务器来进行服务。一种常见的方案是某网点管理其自身的主服务器，而让 ISP 的服务器作为一个备份。一旦配置好了系统，ISP 的服务器就会自动从您的主服务器上下载配置信息。在不给任何一方网点的系统管理员带来任何直接工作负担的情况下，对 DNS 配置所做的更改就能反映在备份服务器上。

不要将您所有的 DNS 服务器放在同一个网络上。当 DNS 停止工作的时候，对于您的用户来说，也就相当于网络停止了工作。将您的 DNS 服务器分散到各处，这样一来，您所得到的就不是一个有单点故障的脆弱系统。经过仔细配置的话，DNS 是相当健壮的。

15.4.2 选择域名

我们以前的建议是域名应该简短并易于键入，而且应该能显示出使用它们的机构的身份。如今的现实却是，所有合适和简短的域名都已经被注册过了，至少在 com 域内是这样。虽然这会让人忍不住要责备这样的抢注行为，但是事实上这些好域名的绝大多数实际都在使用。在 2004 年，超过 60%的注册域名在使用，而历史上看，不到一半的注册域名在投入实际使用。

15.4.3 域名爆炸

DNS 的设计是把一个机构的域名映射到该机构所用的一个名字服务器上。在这种模式下，需要 DNS 能够按世界上的机构数量成比例地扩展。但是如今 Internet 已经成为众多文化的交流渠道，域名正被应用到每一种产品、电影、体育事件、英语名词等上面。例如像 twinkies.com 这样的域名与生产该产品的公司并不（直接）相关，它们只是被用作广告。不清楚 DNS 是否能按这种方式继续扩展下去。这里的实际问题是，DNS 的域名树仅在具有层次结构而非完全平面的情况下才是一种有效的数据结构。随着每个组织都在这棵树的顶级域命名数百乃至数千种产品，那么层次结构就必然会走向末日。

我们真正需要的是一种目录服务，它将品牌和市场营销名字映射到公司机构上，而把 DNS 解放出来，去处理 IP 地址。另一种可能的解决方案是在域名系统上强制推行层次结构，例如 twinkies.hostess-foods.com。不过这种方案永远不会发生——我们在“marketing-domain-name.com”的路上已经走得太远了（译者注：指在顶级域名 com 之后直接冠以营销域名的做法）。

从 DNS 的角度上看，索尼公司使用了正确的方式——其所有产品都是 sony.com 的子域。或许需要多花一两次点击才能找到您想要的产品，但是 DNS 会感激这种层次结构的。

15.4.4 注册二级域名

要获得一个二级域名，必须向适当顶级域的权威机构申请。ICANN 目前正授权若干个代理机构作为其在 gTLD 内注册域名的共享注册方案的一部分。在编写本书英文版的时候，有 500 个注册机构可供选择。访问 www.icann.org 以查看确切的代理清单。

在欧洲注册 ccTLD 域名，可联系地址为 www.centr.org 的欧洲国家顶级域注册委员会（Council of European National Top-level Domain Registries），以确定本地注册机构并申请域名。对于亚太地区，相应的机构是亚太网络信息中心（Asia-Pacific Network Information Center），地址是 www.apnic.net。

要填写完整域名注册表格，您必须为您的域确定一位技术联系人、一位行政联系人和至少两台作

www.TopSage.com

为域服务器的主机。

15.4.5 创建子域

创建一个子域的过程和创建一个二级域的过程类似，只是原来的中心权威机构现在成了本地（或者更精确地说是在您的机构内）的权威机构。具体步骤如下：

- 选择本地环境中惟一的域名；
- 确定两台或更多台主机作为新域的服务器；
- 与父域管理员取得协调。

父域在执行授权之前，应该核查确认子域的名字服务器已启动并在运行。如果服务器不工作，则会产生“残缺授权”，您可能会收到讨厌的电子邮件要求您清除 DNS 操作。15.16.7 节将更详细地解释残缺授权。

15.5 DNS 的工作原理

使用 DNS 的每台主机要么是 DNS 系统的客户机，要么既是客户机又是服务器。如果您不打算运行任何 DNS 服务器，就没有必要阅读以下几节（可直接跳到 15.10 节）。不过，它们会帮您更坚实地理解 DNS 的体系结构。

15.5.1 授权

所有名字服务器都知道根服务器。根服务器又知道 com、org、edu、fi、de 和其他顶级域。顺着这条链进一步往下，edu 知道 colorado.edu，com 知道 admin.com，依此类推。每个区都可以把它对其子域的权威性授权给其他服务器。

让我们看一个真实的例子。假设我们希望从机器 lair.cs.colorado.edu 查询机器 vangogh.cs.berkeley.edu 的地址。主机 lair 询问它的本地名字服务器 ns.cs.colorado.edu 来寻求答案。图 15.1 说明了后面接连发生的事件。

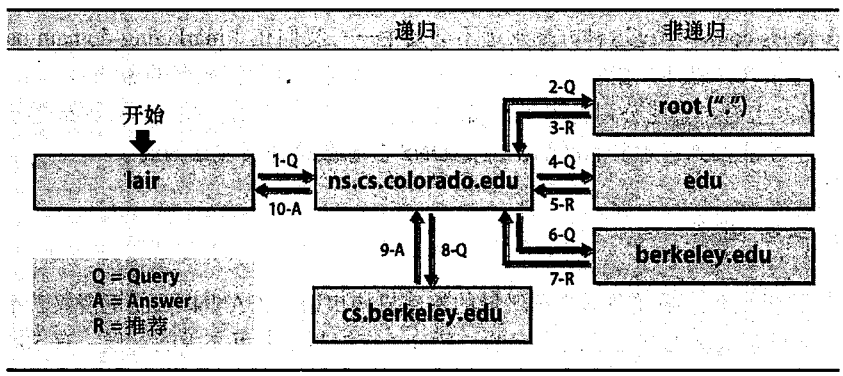


图 15.1 针对 vangogh.cs.berkeley.edu 的 DNS 查询过程

服务器间箭头上的数字显示了事件的次序，字母代表事务的类型（查询、推荐或回答）。我们假设查询之前，除了根服务器的名称和 IP 地址之外没有缓存任何所需要的信息。

本地名字服务器不知道这个地址，而且它不知道有关 cs.berkeley.edu 或 berkeley.edu 甚至 edu 的任

www.TopSage.com

何信息。不过它知道根域的一些服务器，既然它是一台递归服务器，所以它向某个根服务器查询有关 vangogh.cs.berkeley.edu 的信息。

本地名字服务器随后将查询发送到 edu 服务器（还是一样，询问有关 vangogh. cs.berkeley.edu 的信息），得到的反馈是推荐了一些 berkeley.edu 的名字服务器。接着在 berkeley.edu 域中重复该查询。如果 Berkeley 服务器不递归且在缓存中没有答案，则返回的是推荐 cs.berkeley.edu。cs.berkeley.edu 对于要查询的信息来说是权威的，因而返回了 vangogh 的地址。

就在这一眨眼的功夫，ns.cs.colorado.edu 就缓存了主机 vangogh 的地址。它还在服务器上缓存了 edu、berkeley.edu 和 cs.berkeley.edu 的数据。

15.5.2 缓存和效率

缓存（caching）提高了查询的效率：因为映射关系不会经常变动，被缓存的回答几乎总是不用再查的，而且通常是正确的。大多数查询都是针对本地主机的查询，可以快速解析。用户还无意中帮助提高了效率，因为他们重复了许多查询，在查过一次后，以后的就可以不用查而“白得”了。

在很长的一段时间里，缓存机制仅能用于肯定的回答。如果找不到主机名或地址，事实就不被保存。RFC1034 中描述了 DNS 缓存否定答案的方案，但它还不完整，没有得到广泛实现。在 1998 年，RFC2308 定义了更好的方案。该方案在 BIND 8.2 中作为一种可选功能得到实现，而现在在 BIND 9 中则是必选功能。

在欧洲，对 RIPE 根服务器的一项测定显示，60%的 DNS 查询是针对不存在的数据（许多查询把 127.in-addr.arpa 或微软的服务名当作主机名）。沿着 DNS 树进一步往下缓存这类信息将会显著降低根服务器上的负载。

缓存否定回答的机制保存以下类型的回答：

- 没有与要查询的名称相匹配的主机或域；
- 该主机没有要查询的数据类型；
- 询问的服务器无响应；
- 因为网络问题，服务器不可达。

前面两种类型的否定数据，其缓存时间默认为 10 分钟。您可以用 15.7.2 节讨论的 SOA 记录里的一个参数，把这个缓存时间增加到 3 小时，而用一个 BIND 选项则可以增加到一星期。

大多数实现不缓存后面两种否定数据。不过，BIND 确实会将不响应的服务器降格处理，只要有别的选择，那么就不查询它们。如果一个区的所有服务器都没有响应，那么 BIND 都不缓存这样的结果。

非权威性回答可能会被缓存，而权威性的否定回答则必须缓存。BIND 遵守了这些 RFC 给出的规定，但是 Windows 机器似乎在实现 TTL（Time to Live，存活时间）上区别对待，至少对于缓存否定回答来说是这样。它们在查询第一次返回的 NXDOMAIN（无此域）的时候，使用正确的默认值（SOA 记录的参数 *minimum* 的值），以后就将 TTL 重新设置为 15 分钟，让它从缓存里正常超时。

一个名字服务器在响应某条查询时经常会接到多条 DNS 记录。例如，对根域的名字服务器的一次查询将收到一则响应，其中罗列出全部 13 个根服务器。您的服务器要查询哪一个呢？

当几个远程服务器都是某个域的权威服务器，而 BIND 名字服务器必须在这几个服务器中做出选择时，它会首先判断到每个服务器的网络往返时间（Round Trip Time，RTT），然后根据它们的 RTT 将这些服务器分类，并从最快一类中选择一个服务器。在同一类中的服务器则同等对待，轮流使用。

通过给几个 IP 地址（事实上是不同的机器）指派同一个主机名，您可以得到一种原始但有效的负载平衡形式：

www.TopSage.com

TopSage.com

```
www     IN  A   192.168.0.1
        IN  A   192.168.0.2
        IN  A   192.168.0.3
```

像 Yahoo! 或 Google 这样繁忙的 Web 服务器实际上并不是一台单独的机器。它们只是在 DNS 中使用同一个名字[4]。具有相同名字和记录类型的多条记录的名字服务器将所有这些记录返回给客户机，但是按轮流次序。例如，对上述 A 记录第一次查询的轮流次序是 1、2、3，下一次查询是 2、3、1，第三次是 3、1、2，依此类推。

15.5.3 扩展的 DNS 协议

DNS 协议的最初定义要从 20 世纪 80 年代末期开始算起，它使用了端口 53 上的 UDP 和 TCP 协议。UDP 通常用于查询和响应，TCP 用于主服务器和从服务器之间的区传送（zone transfer）。遗憾的是，在所有 UDP 实现中能保证正常工作的最大包长是 512 字节，对于在每个包中必须含有数字签名的一些 DNS 安全新特性（例如，DNSSEC）来说实在是太小了。

512 字节的限制还影响了根服务器的数量和名字。要让所有的根服务器数据能包含在一个 512 字节的 UDP 包中，根服务器的数量只能限制在 13 个，而每个服务器要使用字母表中的单个字母命名。

许多解析器首先发送一条 UDP 查询，如果它们接收到一条被截断的响应，则会用 TCP 重新发送该查询。这个过程绕过了 512 字节的限制，但是效率不高。您或许认为 DNS 应该避开 UDP，总是使用 TCP，但是 TCP 连接的开销要大得多。一次 UDP 名字服务器交换可以短到两个包：一个查询包、一个响应包。一次 TCP 交换则至少包含 7 个包：三次握手初始化 TCP 会话、一个查询包、一个响应包以及最后一次握手来关闭连接。

15.6 DNS 的新特性

DNS 的最新发展前沿都放在了政治领域，而不是技术领域。VeriSign 这家过去曾经垄断域名注册的注册公司，现在负责 com 和 net 域，该公司给这两个域加入了一种通配的地址记录。该记录可以让每个敲错了域名的用户被指到由 VeriSign 的广告商之一所维护的一个网站。这种服务叫做 Site Finder（网站查找）。

Internet 界对这种不公平的事情反响很大，于是 ISC 给 BIND 增加了一个 delegation-only 选项，支持这个选项的 BIND 能够认出这些通配结果，并且返回一个更准确的响应“no such domain（无此域名）”，取代由 VeriSign 提供的地址。这项纠正对于大多数顶级域来说还不错，但不是所有的顶级域都如此，因此后来又增加了一个 exceptions 子句，提供更细致的控制。我们将在 15.11.6 节介绍这些新的 BIND 选项。在经受了一个月的抱怨之后，VeriSign 删除了通配记录，关闭了这项服务。在律师把一切都搞定之后，它们可能还会卷土重来。IETF 可能最终会加紧规范，根本不允许有通配。

在过去的几年中，DNS 在技术上发生了几个明显的变化。本节让您大致了解这些主要变化，并指出详细阐述这些变化的章节。特别而言，IETF 已经从根本上改变了 DNS 相关标准中针对 IPv6 和 DNS 安全的内容，这使得本书上一版中关于这两个专题的内容都完全错了。表 15.4 列出了这些主要的变化，并且指出本书详细介绍它们的章节。

[4] 我们上次查的时候，Google 是 40 万台 Linux 机器（他们不说，但我们用 google 搜了个估计值），而 Yahoo! 由 10 万多台 FreeBSD 机器构成。

www.TopSage.com

表 15.4 DNS 和 BIND 的新发展

章 节	RFC	特 性
15.6	3492	Punycode 国际化的域名
15.6	2671	ENDS0，协议变更和扩展
15.7.2	1996	区变更的异步通知
15.7.8	2317	无类 in-addr 的授权（CNAME 扩展）
15.7.10	2782，3958	用于服务位置的 SRV 记录
15.7.12	–	用于 IPv6 地址的 AAAA 记录（A6 不再使用）
15.7.14	2672-3	被废弃的 DNAME 记录
15.7.14	–	记录用于反向 IPv6 映射的 ip6.arpa 域；ip6.int 被废弃
–	3595，3646	IPv6 支持
15.14.1	1995	增量区传送
15.14.2	2136	动态更新（用于使用 DHCP 的站点）
15.15.3	2845，2930，3645	TSIG/TKEY 事务签名和密钥交换
15.15.4	3225-6，4033-5	DNSSEC、区数据的身份验证和安全性[a]

a. 2004 年完全重做。

这些新特性有些是 IETF 尚未完成标准化的浩大工程。编写这些标准的工作组拥有优秀的文字编写人员，但是缺乏敏锐的代码编写人员，某些最新的规范可能难以或甚至不可能实现。BIND 的当前发布版本（9.4）包括了几乎所有这些新特性。

支持 IPv6 和 DNSSEC 是值得做一点说明的两大类新特性。IPv6 将 IP 地址的长度从 32 位增加到 128 位。IPv6 一旦完全实现，它将会对 Internet 产生巨大的影响。BIND 9 支持 IPv6 中迄今已被标准化的那些部分，但是看起来在本书的生命期内 IPv6 不太可能投入广泛使用。因此，我们将简要介绍 BIND 9 对 IPv6 的支持。本章足以提供给您一般性的了解，但不足以让您将网点迁移到 IPv6 并为它配置 DNS。IPv6 在第 12 章中有更详细的介绍。

DNSSEC 标准给 DNS 数据库及其服务器增加身份验证。它使用公钥加密方法来验证 DNS 数据源以及数据完整性，并使用 DNS 来分发密钥和主机数据。

另外还引入了更简单的身份验证机制，如通过使用“共享秘密（shared secret）”来支持身份验证。不过，共享秘密必须分发到希望相互进行身份验证的每对服务器。尽管这适合于有少数服务器的本地网点，但是不能扩展到 Internet 的层面上。BIND 9 既实现了 DNSSEC 公钥系统，也实现了 TSIG（Transaction Signature，事务签名）共享秘密系统。

从 BIND 9.3 开始包括 DNSSEC 的新规范。不过，因为人们在两年以前就开始实验带签名的区了，所以他们认识到，原来的 DNSSEC 体系不实用。在原来的体系下，一个父区签发一个子区的密钥，两个区都保留该密钥的副本。如果子区想要改变它的密钥，它就不得不要和父亲协商，请求签发新的密钥。这没问题。不过，如果父区想要改变它的密钥，它就会更新所有子区的密钥，而这些密钥既有保存在父区本身的，也有保存在其所有子区里的。这项操作经证明对于像 com 这样大的区不可行，因为在更新期间，肯定有一些子区访问不到。它们的密钥就不能保持同步，造成 DNS 不能核准签名。

目前对此的解决方案是，让每个子区的密钥都只存在于子区里，但是在父区引入一个新的资源记录：DS（delegation signer）。我们将在 15.15.4 节详细介绍 DNSSEC。

国际化域名就是允许使用非英文字符，通过将 Unicode 字符映射回 ASCII 字符的巧妙方法，得以推进了国际化域名的使用。一种称为 Punycode 的系统使用叫做 Bootstring 的算法，能够建立惟一且可逆的映射，参考 RFC3492 了解详情。到 2005 年，域名注册机构已经开始公布 Punycode 域名，大多数浏览器也实现了该系统的某种形式。遗憾的是，同时也清楚地出现了几种和 Punycode 有关的欺骗和安

www.TopSage.com

全问题。此外，国际化域名有效地缩短了DNS名字所允许的最大长度（域名的每个组成部分以及整个域名本身）。

目前的国际化方案避开了一个关键问题，即反别名（antialiasing），因为它非常难解决。亚洲语言的字符和在DNS中表示它们的Punycode编码的Unicode存在不一致性，做到反别名就要解决这种不一致性。如果一个字符的意思可能是3、4、10种不同Unicode之一的话，那么语言专家就必须就翻译标准取得一致的意见，而且在计算机屏幕上显示的字符就必须根据含义的不同而不同。

这3件大事（IPv6、DNSSEC和国际化）中的每一件都明显增加了DNS数据记录的长度，因此DNS更可能碰到UDP分组大小的限制。

在20世纪90年代中期，对DNS协议进行了修订，补充了增量区传送（类似于在旧和新的区文件之间进行一次**diff**操作，这是从Larry Wall的**patch**程序得到的灵感）、异步通知（在主服务器的数据文件已经更新后通知从服务器）和动态更新（用于DHCP主机）。这些变动增加了功能但并没有真正解决基本的传输问题。

在20世纪90年代末期，EDNS0（Extended DNS，version 0，扩展的DNS，版本0）解决了今天的Internet中DNS协议的一些缺点。它能让发送方宣告其重组缓冲区大小、可支持选项和通信协议版本。如果接收方的名字服务器以一条错误消息作为响应，那么发送方就退回去使用原来的DNS协议。BIND 9在服务器和解析器中都实现了EDNS0。

15.7 DNS数据库

一个域的DNS数据库是由这个域的主名字服务器的系统管理员所维护的一个文本文件集合。这些文本文件经常称为区文件（zone file）。它们包含两种类型的项：分析器命令（像$ORIGIN和$TTL）和“资源记录”，有时也称为RR（Resource Records）。只有资源记录才是数据库的真正部分，分析器命令只是为记录输入提供的一些捷径。

我们从描述DNS资源记录开始本节的内容，资源记录在RFC 1035、1183、1876、2230、2782、2930、3596和3658中定义。我们到15.7.16节再讨论分析器命令。

15.7.1 资源记录

DNS层次结构中的每一个区（zone）都有与之相关的一组资源记录（这组记录可能为空）。资源记录的基本格式是：

```
[name] [ttl] [class] type data
```

各字段间由空白（Tab或空格）分隔，字段可以包含表15.5中列出的特殊字符。

表15.5 RR中使用的特殊字符

字 符	含 义
;	引入注释
@	当前的域名
()	允许数据跨行
*	通配符[a]（仅由name字段使用）

a. 关于一些要注意的语句，请参考15.7.7节。

name（名字）字段表示该记录所描述的实体（通常为一个主机或域）。如果几个连续的记录涉及

www.TopSage.com

同一个实体，则第一条记录之后可以省略这个名字。如果出现这个字段，则必须从第一列开始。

名字可以是相对名或全名。全名是以点（.）结尾的完整域名。软件在内部仅处理全名，它把当前域和一个点追加到没有以点结尾的任何名字之后。这一特性能让名字更短一些，但是也容易出错。

例如，在 cs.colorado.edu 域中，名字“anchor”将被解释为“anchor.cs.colorado.edu.”。如果输入的名字为“anchor.cs.colorado.edu”，结尾没有点则意味着这是一个相对名，要给它加上默认的域，结果就变成域名“anchor.cs.colorado.edu.cs.colorado.edu.”。这是一种十分常见的错误。

ttl（time to live，存活时间）字段以秒为单位指定数据项可被缓存并且仍被认为有效的时间长度。除了在根服务器的线索文件中之外，该字段经常被省略。它默认取该区的数据文件开头的`$TTL`指令设置的值。在 BIND 9 中，要求用`$TTL`指令。如果在 BIND 8 中没有`$TTL`指令，*ttl* 默认为区的 SOA 记录中给各个区设置的值。

ttl 将存活时间参数的值增加到大约一周将明显减少网络流量和 DNS 负载。不过，一旦记录已经缓存在本地网络之外，您就无法迫使它们被丢弃。如果您准备大规模地重新对网络编号，可以降低$TTL 的值（例如，降为 1 个小时），以使缓存在 Internet 上别处的老记录迅速过期。请参考第 17 章获得有关 NIS 的更多信息。

有些站点将面向 Internet 的服务器的 DNS 记录里的 TTL 值设得比较低，这样一来，如果服务器遇到问题（网络故障、硬件故障、拒绝服务攻击等），系统管理员可以通过改变该服务器的“名字到 IP 地址”映射做出响应。因为原来的 TTL 比较低，那么新设的值会迅速传播开来。例如，google.com 这个名字的 TTL 是 5 分钟，但是 Google 的名字服务器的 TTL 却是 4 天（345 600 秒）。

```
google.com.      300     IN A  216.239.37.99
google.com.      345 600 IN NS nsl.google.com.
nsl.google.com. 345 600 IN A  216.239.32.10
```

我们使用 **dig** 命令（**dig @ns1.google.com google.com**）去重新获得这些数据这里的输出结果经过了裁剪。

BIND 9 实施了一种称为 TTL harmonization（TTL 协调）的概念，要求在一个 RR 集（RRset）里的所有记录（也就是说，针对一个节点的所有同类型记录）都有相同的 TTL。实际使用的 TTL 值是该“节点/类型”对的第一个资源记录的 TTL 值。

class（类）指定网络类型。这个字段可以识别的值有 3 个：

- IN 对应 Internet；
- HS 对应 Hesiod；
- CH 对应 ChaosNet。

这个 *class* 的默认值是 IN。虽然它是一个默认值，但还是经常在区域数据文件中显式地指定。由 MIT 开发的 Hesiod 是一个建立在 BIND 上的数据库服务。ChaosNet 是一种已经被废弃的网络协议，以前由 Symbolics Lisp 机器使用。

现如今，在 ChaosNet 类中正常情况下会把两种标识数据给隐藏起来：名字服务器软件的版本号、服务器在运行的主机的名字。这些重要数据像 15.8.2 节介绍的那样，可以用 **dig** 提取出来。系统管理员利用名字服务器的版本号来找出需要升级的服务器，他们还可以使用主机的标识来对通过 anycast 路由复制的名字服务器进行调试。原来 BIND 实现里有一种功能（有人可能说是“巧用”），即让这些信息能通过 CH 类得到，但是现在已经由 IETF 把它作为 DNS 正当的一部分写入标准[5]。

[5] 遗憾的是，对于数据应该放在哪个名字下却存有一些争议。它应该是 version.bin？是 hostname.bind？还是 id-server 或者……？

www.TopSage.com

虽然定义了许多种不同类型的DNS记录，但是常用的不到10种，IPv6又增加了几种。我们将资源记录划分为4组：

- 区记录——标识域和它们的名字服务器；
- 基本记录——将名字映射到地址并路由邮件；
- 安全记录——向区文件添加身份验证和签名；
- 可选记录——提供有关主机或域的额外信息。

data（数据）字段的内容取决于记录类型。表15.6列出了几种不同的记录类型。

表15.6 记录的类型

	类 型	名 称	功 能
区记录	SOA	Start Of Authority	定义一个DNS区
	NS	Name Server	标识区的服务器，授权子域
基本记录	A	IPv4 Address	名字到地址的转换
	AAAA[a]	IPv6 Address	名字到IPv6地址的转换
	PTR	Pointer	地址到名字的转换
	MX	Mail Exchanger	控制邮件的路由
安全记录	DS	Delegation Signer	签发子区的密钥的密钥散列
	DNSKEY	Public Key	DNS名称的公钥
	NSEC	Next Secure	和DNSSEC一起使用，用于否定回答
	RRSIG	Signature	带签名、经过身份验证的资源记录集
可选记录	CNAME	Canonical Name	主机的昵称或者别名
	LOC	Location	地理位置和范围
	SRV	Services	提供知名服务的位置
	TXT	Text	注释或者非键入的信息

a. AAAA和A6这两种IPv6地址记录已经在IETF内争论过好几年了。最终AAAA获胜，又从废弃返回到标准。A6现在被标为实验性的记录。

有些记录类型是废弃的、实验性的、或者没有得到广泛使用。请参阅BIND文档了解记录的完整名单。大多数资源记录要手工维护（通过编辑文本文件），但是安全记录要求加密处理，所以必须用软件工具管理。这些记录将在介绍DNSSEC的15.15.4节讨论。

资源记录的次序几乎可以是任意的。一个区的SOA记录应该在最前面，但是这个要求现在也放宽了。NS记录通常紧跟在SOA后面。每个主机的记录通常放在一起。按照*name*字段进行排序是一种常见的做法，有些站点用IP地址来排序，这样一来，就更容易找到没有用到的地址。

伴随我们对每种资源记录的详细介绍，我们会考察一些样例，它们取自于cs.colorado.edu的数据文件。默认的域都是“cs.colorado.edu.”，所以“anchor”所指的主机实际上代表“anchor. cs.colora do.edu.”。

15.7.2 SOA记录

一个SOA记录标记出一个区，即位于DNS域名空间中相同位置的一组资源记录的开始位置。DNS树的这个节点也称为一个授权点或区分割。正如我们将在15.7.4节详细讨论的那样，用于一个DNS域的数据通常至少包括两个区：一个区用来把主机名转换成IP地址，另外一些区进行反向的映射。DNS树有一个由名字组成的正向分支和一个由IP组成的反向分支。

每个区仅有一个SOA记录。该区一直延伸到遇见另一个SOA记录为止。SOA记录包括区的名字、一个技术联系人和各种不同的超时值。看一个例子：

www.TopSage.com

大家网 TopSage.com

```
; Start of authority record for cs.colorado.edu
@           IN  SOA  ns.cs.colorado.edu. hostmaster.cs.colorado.edu. (
                     2004111300     ; Serial number
                     7200           ; Refresh    (2 hours)
                     1800           ; Retry      (30 minutes)
                     604800         ; Expire     (1 week)
                     7200 )         ; Minimum    (2 hours)
```

这里，*name* 字段包含符号@，这是当前区名的简写。本例中可以用“cs.colorado.edu.”代替它。@的值是在 **named.conf** 文件中的 zone 语句中指定的名字，它可以在这个区文件中用$ORIGIN 分析器指令进行更改（参考 15.7.16 节）。

本例中没有 *ttl* 字段。*class* 是代表 Internet 的 IN，记录类型是 SOA，剩下的几项构成 *data* 字段。

“ns.cs.colorado.edu.”是该区的主名字服务器。

“hostmaster.cs.colorado.edu.”是技术联系人的电子邮件地址，它使用“*user.host.*”的格式，而不是标准的“*user@host*”格式。如果需要发送邮件给域管理员，则只需将第一个点替换成@，并删除末尾的点即可。各网点经常使用像 admin 或 hostmaster 这样的别名来代替一个真实的登录名。负责 hostmaster 的系统管理员可能会换人，因此，当您需要更新联系人的时候，修改 **aliases** 文件里的一项（参考 18.4 节）要比修改所有的区文件容易得多。

小括号能使 SOA 记录跨越好几行。括号的位置在 BIND 4 或 8 中不是任意的——我们试图通过在联系地址之前分割首行来将这一行缩短，但是后来 **named** 却无法识别这个 SOA 记录。某些实现仅识别 SOA 和 TXT 记录中的小括号。BIND 9 有一个更好的分析器，可以将小括号放在任何位置。

第一个数值参数是这个区配置数据的序列号。这个序列号供从服务器判断何时获取新数据之用。它可以是 32 位的任何整数，每当更改区的数据文件时，都应该增加这个序列号的值。许多站点把文件的修改日期编成序列号。例如，2004111300 就是 2004 年 11 月 13 日对该区做首次更改。

序列号不必连续，但是必须单调递增。如果您偶然在主服务器上设置了一个非常大的数，并将这个值传送到了从服务器，那么即使纠正主服务器上的序列号也不会起作用。只有当主服务器上的序列号大于从服务器的序列号时，从服务器才请求新数据。

有 3 种方法可以解决这个问题，在 BIND 9 里只有前两种方法能起作用。

- 解决该问题的一种方法是利用序列号所在的顺序空间的属性。这个过程包括在序列号上叠加一个大数值（2^{31}），让所有从服务器传送数据，然后将序列号设置为您想要的数。RFC1982 解释了顺序空间。
- 解决该问题的一个隐蔽但更繁琐的途径是更改主服务器上的序列号，杀死从服务器进程，删除从服务器上的备份数据文件，因而强制它们从主服务器重新装载，并重新启动从服务器。
- BIND 4.9 和 BIND 8 包含一个窍门，可以让您以一定刷新时间间隔将序列号设置为零，然后重新启动编号过程。零总会导致重新装载，因此在每个从服务器已经重新装载序列号为零的区后，不要忘记将序列号设置为一个真实值。

更改数据文件但是忘记更新序列号是一种常犯的错误。作为惩罚，名字服务器将不把更新传送到从服务器。

SOA 记录中接下来的 4 项是以秒为单位的超时值，它们控制数据在全球范围 DNS 数据库的不同点上的缓存时间。这些时间值能在后面加上 m、h、d 或者 w，分别表示以分钟、小时、天或者星期为单位。例如，1h30m 代表 1 小时 30 分钟。超时值体现出了在效率（使用一个旧值比取回一个新值代价要小）和准确性（新值更准确）之间的一种折中。

下面再给出同一个 SOA 记录的例子，只是为了让您不必再翻页了。

www.TopSage.com

```
; Start of authority record for cs.colorado.edu
@           IN  SOA  ns.cs.colorado.edu. hostmaster.cs.colorado.edu. (
                          2004111300     ; Serial number
                          7200           ; Refresh    (2 hours)
                          1800           ; Retry      (30 minutes)
                          604800         ; Expire     (1 week)
                          7200 )         ; Minimum    (2 hours)
```

第一个是 *refresh*（刷新）超时，指定从服务器应该多长时间与主服务器进行核对，以查看这个区所配置的序列号是否已经有了变化。每当区有变化时，从服务器必须更新它们的区数据副本。从服务器要比较序列号，如果主服务器的序列号大，那么就要求一次区传送，更新数据。*refresh* 超时的常用值可以为 1~6 个小时（3 600~21 600 秒）。

每当区有变化时，除非在配置文件中特意关闭 notify 参数，否则 BIND 服务器都会通知它们的从服务器，而不只是被动地等待从服务器超时。理解这种通知的从服务器将立刻行动，刷新自身。更新通知有可能因为网络拥塞而丢失，所以应该设置一个合理的刷新超时值。

如果一个从服务器试图检查主服务器的序列号，但是主服务器不响应，从服务器会在经过超时时间 *retry* 后再次尝试。我们的经验表明 20~60 分钟（1200~3600 秒）是个比较合适的值。

如果一个主服务器停机了很长一段时间，从服务器多次尝试刷新数据但总是失败。每个从服务器应该最终判断得知主服务器永远无法反应，它的数据肯定是过时的。*exprire* 参数决定了从服务器在没有主服务器的情况下权威地持续提供域数据服务的时间长短。如果主服务器停机一周，系统应该能够维持下来，因此这个参数应该设置为稍长的值。我们推荐一周到一个月。

SOA 记录中的 *minimum* 参数设置被缓存的否定回答的存活时间[6]。肯定回答（也就是说真实记录）的默认值是在区文件的开头处，用 `$TTL` 指令设定的。`$TTL` 的经验值为几个小时到几天，*minimum* 值为两小时或者一天。`$TTL` 的值必须大于或者等于 *minimum*。

`$TTL`、*expire* 和 *minimum* 参数最终迫使使用 DNS 的每个人放弃旧数据值。DNS 最初的设计依赖于主机相对稳定并且不经常更改这一事实。不过，DHCP 和移动主机已经改变了这一规则。BIND 通过提供动态更新和增量区传送机制（参考 15.14 节），努力地应付这种情况。有关 TTL 和一种称为 TTL harmonization（TTL 协调）的概念的更多信息，请参考 15.7.2 节。

15.7.3 NS 记录

NS（名字服务器）记录标识一个区的权威服务器（也就是说，所有主服务器和从服务器），并把子域授权给其他机构。NS 记录通常跟在 SOA 记录后面。

格式为：

```
zone [ttl] IN NS hostname
```

例如：

```
cs.colorado.edu.    IN  NS    ns.cs.colorado.edu.
cs.colorado.edu.    IN  NS    anchor.cs.colorado.edu.
cs.colorado.edu.    IN  NS    ns.cs.utah.edu.
```

因为区的名字与这些 NS 记录之前的 SOA 记录中的 *name* 字段相同，所以可以留为空白。因此，

[6] 在 BIND 8.2 之前，*minimum* 参数设定的是资源记录默认的存活时间。它包含在每条记录中，用于让非权威的服务器上被缓存的记录超期作废。

www.TopSage.com

下面这 3 行和紧跟在 cs.colorado.edu 的 SOA 记录之后的行是等价的：

```
IN  NS    ns.cs.colorado.edu.
IN  NS    anchor.cs.colorado.edu.
IN  NS    ns.cs.utah.edu.
```

为了能对外界可见，cs.colorado.edu 的每个权威名字服务器应该既要在 cs.colorado.edu 的区文件中列出，也要在父区 colorado.edu 的文件中列出。仅缓存服务器不能是权威的，不用列出它们。NS 记录中没有参数指定服务器是主服务器还是从服务器。该信息在名字服务器的配置文件中指定。

当 BIND 希望把对区的更改通知发送出去时，它使用一个区的 NS 记录来识别从服务器。在父区（colorado.edu）内的那些相同的 NS 记录定义 cs 子域，并把对该域的权威授权给计算机科学系的名字服务器。如果父区中的名字服务器列表没有与那些在该区本身内部的列表一起保持更新，那么添加的所有新服务器都变成了“秘密的服务器”，不能用来回答来自外界的查询。有时这种情况是因为设计的目的，有时则是因为疏忽。只要父区至少有子区的一个有效 NS 记录，那么这就不是个严重的错误。

快速查看我们自己的授权情况，发现 edu 域不知道的 colorado.edu 的一个主要服务器。照我们说的而不是我们做的做：用 **dig** 检查授权情况，以确认它们指定了一个正确的服务器集合（参见 15.16.6 节）。有关授权的更多信息，请参考 15.7.17 节。

15.7.4 A 记录

A（地址）记录是 DNS 数据库的核心。它们提供了以前在**/etc/hosts** 文件中指定的主机名到 IP 地址的映射。一个主机必须为它的每个网络接口得到一条 A 记录。格式是：

```
hostname [ttl] IN A ipaddr
```

例如：

```
anchor          IN A 128.138.243.100
```

具有多个网络接口的机器可以让其所有接口关联到同一个主机名上，也可以每个接口都有各自的主机名。

15.7.5 PTR 记录

PTR（指针）记录执行从 IP 地址到主机名的反向映射。与 A 记录一样，主机必须给每个网络接口一条 PTR 记录。但是在我们描述 PTR 记录之前，我们需要岔开话题一会儿，谈谈称为 in-addr.arpa 的一个特别的顶级域。

完整的主机名可以看成是一种记法，在这种记法中“最大的部分”在右边。例如，在域名 anchor.cs.colorado.edu 中，anchor 在 cs 中，cs 在 colorado 中，而 colorado 在 edu 中。另一方面，IP 地址的“最大的部分”在左边。在地址 128.138.243.100 中，主机 100 在子网 243 上，该子网又是网络 128.138 的一部分。

创建 in-addr.arpa 能让一组软件模块和一棵域名树既可以把 IP 地址映射为主机名，也可以把主机名映射为 IP 地址。in-addr.arpa 下的域的命名类似于反向字节顺序的 IP 地址。例如，我们的 243 子网的区是 243.138.128.in-addr.arpa。

PTR 记录的通用格式是：

```
addr [ttl] IN PTR hostname
```

www.TopSage.com

例如，在243.138.128.in-addr.arpa区中与前面anchor的A记录相对应的PTR记录为：

```
100          IN  PTR  anchor.cs.colorado.edu.
```

名字100没有以一个点结尾，因此它是相对的。但是相对于什么呢？不是“cs.colorado.edu.”。要让这条记录准确，默认域必须是“243.138.128.in-addr.arpa.”。

和这个例子一样，您可以把每个子网的PTR记录放入它们自己的文件中来设置域。在名字服务器的配置文件中，设置与这个文件相关的默认域。做反向映射的另一种方法是包括类似：

```
100.243      IN  PTR  anchor.cs.colorado.edu.
```

的带有一个默认域138.128.in-addr.arpa的记录。某些站点把所有反向记录放到同一个文件中，并使用$ORIGIN指令来指定子网。注意主机名anchor.cs.colorado.edu必须以一个点结尾，以免把138.128.in-addr.arpa追加到它的名字后面。

因为cs.colorado.edu和243.138.128.in-addr.arpa是DNS域名空间的不同区域（region），所以它们构成两个独立的区（zone）。每个区必须有它自己的SOA记录和RR。除了为每个真实网络定义一个in-addr.arpa区之外，还需要一个管理环回（loopback）网络127.0.0.0的区。

如果子网在字节边界上划分，那么这一切都工作良好。但是如何处理像128.138.243.0/26这样的子网的反向映射呢？RFC2317中定义了一种很好的方法，利用CNAME资源记录来完成这一功能，参见15.7.8节。

PTR记录提供的反向映射能够为任何对网络入流量进行认证的程序所使用。例如，如果在用户的~/**.shosts**文件中列出了来源机器的名字，那么**sshd**可以允许无密码的远程登录。当目标主机接收到一个连接请求时，它只知道源机器的IP地址。它使用DNS把IP地址转换成主机名，然后将该主机名与相应文件作比较。**netstat**、**tcpd**、**sendmail**、**sshd**、X Windows和**ftpd**都使用反向映射从IP地址获得主机名。

A记录与它们对应的PTR记录相匹配很重要。两者不匹配或遗漏PTR记录会导致身份验证失败，这可以使系统缓慢得如同爬行一般。这个问题本身非常令人讨厌，它还有助于造成对任何需要反向映射以匹配A记录的应用程序遭到拒绝服务攻击。

15.7.6 MX记录

电子邮件系统使用邮件交换记录来更有效地路由邮件。MX记录抢占了消息的目的地址，在大多数情况下，它将消息指向接收方网点的邮件枢纽主机，而不是接收方自己的工作站。有关电子邮件的更多信息，请参考第18章。

MX记录的格式为：

```
name [ttl] IN MX preference host ...
```

以下是两个例子，一个用于除非停机否则自己接收邮件的主机，另一个用于根本不接收邮件的主机：

```
piper       IN  MX   10 piper
            IN  MX   20 mailhub
            IN  MX   50 boulder.colorado.edu.
xterm1      IN  MX   10 mailhub
            IN  MX   20 anchor
            IN  MX   50 boulder.colorado.edu.
```

先尝试优先值较低的主机：0最先，65 535最后。在这个例子中，如果mailhub可以访问到，那

www.TopSage.com

么地址为 bob@xterm1 的邮件将被发送到 mailhub，第二选择是发送到 anchor，如果 mailhub 和 anchor 都停机了，则发送到 boulder。注意 boulder 的域名必须是完整的，因为它不是默认域的成员（这里是“cs.colorado.edu.”）。

优先值和主机的列表都可以在同一行，但是独立的行更易于阅读。在优先值之间保留数值上的“间隔”，以便于在需要插入一个新的目的地址时不用重新编号。

MX 记录可用于许多场合：

- 当有中央邮件枢纽时；
- 目标主机停机时；
- 目标主机从 Internet 不可达时；
- 目标主机不支持 SMTP 时；
- 本地系统管理员比和我们通信的人更清楚邮件应被发送到什么位置时。

在第一种情况下，邮件被路由到邮件枢纽，它是多数用户阅读邮件的机器。在第二种情况下，邮件被路由到附近的主机，并在目标主机开机后转发给目标主机。

没有直接连接到（公共）Internet 上的主机不能有 A 记录，但是它们可以有 MX 记录。这种只有 MX 记录的主机可能是在防火墙之后的机器，域名由 ISP、托管服务、或者不是一直开机的机器来代管。**sendmail** 无法直接与目标主机连接，但是它能够通过连接到目标主机的 MX 主机之一，将邮件发到离目标地址更近些的地方。

使用 MX 记录最后也是最重要的原因是本地的系统管理员可能比和您通信的人更了解邮件的结构。他们需要最终确定您的网点如何传输邮件流。

每台让外界知道的主机都应该有 MX 记录。对于次要主机而言，一两条备选的记录就足够了。主要的主机应该有多条记录。例如，下面的一组记录可能适合于每台主机都发送和接收自己邮件的地方：

- 一条 MX 记录用于主机自身，作为首选；
- 部门的邮件枢纽主机作为次选；
- 本域或者父域的中央邮件枢纽主机作为备份。

user@domain 域本身应该有一条到邮件枢纽机器的 MX 记录，以便于发送到 *user@domain* 的邮件能被收到。当然，这样的配置确实要求用户名在域内的所有主机上都是惟一的。例如，为了能向 evi@cs.colorado.edu 发送邮件，我们需要一台名为 cs 的机器，或在 cs.colorado.edu 的 MX 记录，或者可能两者都要。

```
cs          IN  MX  10 mailhub.cs.colorado.edu.
            IN  MX  20 anchor.cs.colorado.edu.
            IN  MX  50 boulder.colorado.edu.
```

替另一台主机接收邮件的机器必须在它的 **sendmail** 配置文件中列出另一台主机，有关 **sendmail** 的 `use_cw_file` 功能和文件 **local-host-names** 的讨论，请参考 18.9.2 节。

有时也可在 DNS 数据库中看到带有通配符的 MX 记录：

```
*           IN  MX  10 mailhub.cs.colorado.edu.
```

乍一看，该记录看起来能够节省许多录入工作，并为所有主机添加了一个默认 MX 记录。但是通配符记录不会像您推测的那样工作。它们匹配了资源记录的 *name* 字段中任何还没有在另一个资源记录中作为显式名称列出的内容。

因此，无法用星号为所有的主机设置一个默认值。但相反，可以用它来为不是您的主机的名字设置默认值。这一设置会让发送到您的邮件枢纽的许多邮件被拒收，因为与星号匹配的主机名实际上不属于您的域。因此要避免使用带有通配符的 MX 记录。

www.TopSage.com

大家网 TopSage.com

15.7.7 CNAME 记录

CNAME 记录为主机多分配几个名字。通常，这些昵称要么用来将某项功能与主机关联起来，要么用来缩短一个长主机名。真实的名称有时称为规范名称（canonical name）（因而叫“CNAME”）。

看几个例子：

```
ftp       IN  CNAME  anchor
kb        IN  CNAME  kibblesnbits
```

CNAME 记录的格式为：

```
nickname [ttl] IN CNAME hostname
```

当 DNS 软件遇到一条 CNAME 记录时，它就会停止对该昵称的查询，并切换到真实的名称。如果主机有 CNAME 记录，该主机的其他记录（A、MX、NS 等）必须引用其真实名称，而不是昵称[7]。例如，下面几行都是对的：

```
colo-gw         IN  A       128.138.243.25
moogie          IN  CNAME   colo-gw
www             IN  CNAME   moogie
```

不过，在这个例子里，为 www 或者 moogie 分配地址或者邮件优先级（用一个 A 或者 MX 记录）都会出错。

在 BIND 中，CNAME 记录可以有 8 层嵌套。也就是说，一条 CNAME 记录可以指向另一条 CNAME 记录，第 2 条 CNAME 又可以指向第 3 条 CNAME，依此类推，最多 7 次，第 8 次的目标必须是真实的 A 记录。

通常您可以给主机的真实名字及其昵称都用 A 记录，从而彻底避免用 CNAME 记录。

15.7.8 巧用 CNAME

CNAME 还用于将 DNS 的现有含义引申为支持不在字节边界上划分子网的网络的反向区。在 CIDR 寻址普及之前，多数子网划分是在字节边界或同一个机构内完成的，反向授权也容易管理。例如，如果 B 类网络 128.138 被划分成一组类似 C 类的网络，那么每个子网会整齐地提供 in-addr.arpa 域的成分。243 子网的反向区是 243.138.128.in-addr.arpa。有关 CIDR 的更多信息，请参考 12.4.4 节。

如果对 243 子网进一步划分，例如分成 4 个/26 网络，会怎么样呢？如果所有 4 个部分都分配给同一个机构，那实际上没有问题。这 4 个子网仍然可以共享一个文件，这个文件包含它们所有的 PTR 记录。不过，如果 243 子网分配给一个 ISP，该 ISP 希望把每个子网授权给不同的客户，那么就必须进行更复杂的处理。ISP 要么必须替每个客户代为维护反向记录，要么必须想办法取出 IP 地址的第 3 个八位组（这里是 243），并将其分割成可以独立授权的 4 个不同部分。

当行政边界落在字节中间时，就必须圆滑一些。还必须与上下级域紧密配合工作。技巧是：对于正常 in-addr.arpa 区中每个可能的主机地址，添加一个 CNAME，把查询转向由相应的子网所有者控制的区。这种方案会使父区上的区文件变得杂乱，但是它的确能够让您把权限授权给每个子网的真实用户。

[7] CNAME 的规则明显为 DNSSEC 放宽了，DNSSEC 给每个 DNS 资源记录加上了数字签名。CNAME 的 RRSIG 记录用于指明昵称。

www.TopSage.com

这里是该方案的细节。父机构（在我们的情况下是 ISP）用一个代表子网的额外虚拟部分（用点分割的部分）为每个可能的 IP 地址创建 CNAME 记录。例如，在刚才描述的/26 情况中，地址的第 1 个四分之一有“0-63”部分，第 2 个四分之一有“64-127”部分，依此类推。看起来就像这样：

```
$ORIGIN 243.138.128.in-addr.arpa.
1           IN  CNAME  1.0-63
2           IN  CNAME  2.0-63
...
63          IN  CNAME  63.0-63
64          IN  CNAME  64.64-127
65          IN  CNAME  65.64-127
...
```

要将反向区域的“0-63”部分授权给已经分配到该子网的客户，需要添加下列 NS 记录：

```
0-63        IN  NS    ns1.customer1.com.
0-63        IN  NS    ns2.customer1.com.
...
```

customer1.com 站点将有一个包含 0-63.243.138.128.in-addr.arpa 区的反向映射的区文件。
例如：

```
1           IN  PTR   host1.customer1.com.
2           IN  PTR   host2.customer1.com.
...
```

通过添加这个额外部分，我们创建了一个新“分割”，并按分割执行授权。例如，当某人查询对 128.138.243.1 的反向映射时，1.243.138.128.in-addr.arpa 上的 CNAME 记录把这次搜索重新聚焦到 1.0-63.243.138.128.in-addr.arpa 上，而这个名字由客户控制。

客户的文件很简单，只是 ISP 必须要处理复杂的配置。但是事情可能变得更加复杂。customer1 自身可能是 ISP，它希望进一步划分它的地址。但是没有关系：BIND 最多支持有 8 个链接的 CNAME 链，因为一个字节只有 8 位，我们永远不可能用完。RFC 中不鼓励但不禁止使用 CNAME 链，因为 CNAME 链上的每个环节会让查询追踪到下一个环节，并发起一条到目标的新查询，因此会减缓域名的解析速度。

在出现 CNAME 巧用法的早期，BIND 的指令系统中加入了$GENERATE 命令（参见 15.7.16 节）来帮助在父区中创建资源记录。例如，要生成第一个子网的记录，采用下面这几行就足够了：

```
$ORIGIN 243.138.128.in-addr.arpa.
$GENERATE 0-63 $ CNAME $.0-63
0-63        NS  ns1.customer1.com.
0-63        NS  ns2.customer1.com.
```

在$GENERATE 命令中的$从 0 一直到 63，创建 64 个不同的 CNAME 记录。其他 3 个/26 网络也作类似处理。

15.7.9 LOC 记录

LOC 记录（LOC 记录在 RFC1819 中定义）描述 DNS 对象的地理位置和物理上的大小（直径）。LOC 记录目前不影响 Internet 的技术操作，没有任何查找它们的标准软件。不过，针对这项信息，建议了若干种有意思的潜在用途，包括路由跟踪和优化、自动映射和网络搜索。LOC 记录在 RFC 1819 中定义。

www.TopSage.com

格式为：

```
name [ttl] IN LOC lat lon [alt [size [hp [vp]]]]
```

lat（纬度）和 *lon*（经度）用以空格分隔的度、分和秒表示，后面跟着 N（北）、S（南）、E（东）或 W（西）。秒可以省略；如果省略了秒，分也可以省略。

其他字段都用厘米（没有后缀）或米（*m*）为单位指定。*alt*（高度）是该对象的海拔高度，*size*（尺寸）是该对象边界范围的直径，*hp*（水平精度）是测量的水平精度，*vp*（垂直精度）是测量的垂直精度。尺寸的默认值是 1 米，水平和垂直精度默认值分别是 10 米和 10 公里。

这里是取自加利福尼亚州圣地亚哥 caida.org 的一个例子：

```
caida.org.          IN  LOC  32 53 01 N 117 14 25 W 107m 30m 18m 15m
```

CAIDA（Cooperative Association for Internet Data Analysis）编写的许多图形可视化工具要求纬度和经度数据，并鼓励网点在 DNS 中包括这些数据。不过，如果您很多疑，又运行着一个知名度很高的服务器或 ISP，那么您可能不希望让一般公众知道您机器的准确位置。在这种情况下，我们推荐您使用非精确值。它们仍然对进行网络搜索的人有价值，但还是提供了一定的匿名性。

15.7.10 SRV 记录

SRV 记录指定域内服务的位置。例如，SRV 记录允许您直接查询一个远程域，并询问其 FTP 服务器的名字。直到现在，您大多得去猜这个名字。为了联络远程域的 FTP 服务器，您只得希望远程系统的管理员已经遵守了目前的习惯，向他们服务器的 DNS 记录中为“ftp”添加一条 CNAME 记录。

对于这种应用，SRV 记录比 CNAME 更有意义，当然是系统管理员用来为移动服务和控制服务使用的一种更好途径。不过，客户机必须显式地查找并分析它们，因此还要等一段时间才能感觉到它们的实际效果。Windows 对它们的用途做了扩展。

SRV 记录带有能让本地 DNS 管理员控制并平衡外界连接负载的字段，这和一般的 MX 记录类似。格式为：

```
service.proto.name [ttl] IN SRV pri wt port target
```

其中，*service*（服务）是在 IANA 分配的编号数据库中定义的一种服务，*proto* 不是 tcp 就是 udp，*name* 是 SRV 记录引用的域，*pri* 是 MX 风格的优先级，*wt* 是用于几个服务器之间平衡负载的权重，*port* 是服务运行的端口，*target* 是提供服务的服务器的主机名。目标的 A 记录通常是自动返回的对 SRV 查询的回答。*wt* 参数为 0 意味着不需要作特别的负载平衡。目标为“.”意味着该站点没有运行这种服务。

这里是一个例子，它取自 RFC2052（在这个 RFC 中定义了 SRV），并针对 cs.colorado.edu 域做了调整：

```
_ftp._tcp          SRV  0  0  21  ftp-server.cs.colorado.edu.

; don't allow finger anymore (target = .)
_finger._tcp       SRV  0  0  79  .

; 1/4 of the connections to old box, 3/4 to the new one
_ssh._tcp          SRV  0  1  22  old-slow-box.cs.colorado.edu.
                   SRV  0  3  22  new-fast-box.cs.colorado.edu.

; main server on port 80, backup on new box, port 8000
```

www.TopSage.com

```
_http._tcp          SRV  0  0  80  www-server.cs.colorado.edu.
                    SRV  10  0  8000  new-fast-box.cs.colorado.edu.

; so both http://www.cs.colo... and http://cs.colo... work
_http._tcp.www      SRV  0  0  80  www-server.cs.colorado.edu.
                    SRV  10  0  8000  new-fast-box.cs.colorado.edu.

; block all other services (target = .)
*._tcp              SRV  0  0  0  .
*._udp              SRV  0  0  0  .
```

这个例子演示了权重参数（用于 SSH）和优先级参数（HTTP）的用法。两个 SSH 服务器都得到使用，两者分担任务。只有在主服务器无法使用时，才使用备份 HTTP 服务器。**finger** 服务没有包括在其中，也不包括没有明确提到的其他服务。**finger** 守护进程没有出现在 DNS 中的事实并不意味着它不在运行，而只是您不能通过 DNS 找到它。

WKS（众所周知的服务）是早期的与服务相关的 DNS 记录，它并不流行。它列出了一个特定主机所提供的服务，而不是把您引向为某个域提供一种特定服务的主机。WKS 看起来没有多少用途，还被认为是一种对安全的危险。它没有得到广泛使用。

15.7.11 TXT 记录

TXT 记录把任意文本添加到一个主机的 DNS 记录中。例如，我们有一个标识我们站点的 TXT 记录：

```
IN        TXT  "University  of  CO,  Boulder  Campus,  CS  Dept"
```

这个记录紧跟在“cs.colorado.edu.”区的 SOA 和 NS 记录之后，所以从它们那里继承了 name 字段。

TXT 记录的格式为：

```
name [ttl] IN TXT info …
```

所有的 *info*（信息）项都必须用引号括起来。您可以使用一个由引号括起来的字符串或多个分别用引号括起来的字符串。确认引号要成对——遗失一个引号将对 DNS 数据造成严重破坏，因为遗失的引号和出现引号的下一个位置之间的所有记录都会莫名其妙地消失。

许多系统管理员使用 TXT 记录发布其站点上的机器名，只有这些机器才是合法的电子邮件来源。别的一些站点使用所谓 SPF（Sender Policy Framework，发件方政策框架）记录来帮助辨别和删除垃圾邮件。SPF 的信息保存在 TXT 记录的 *info* 部分里；SPF 没有单独的 DNS 记录。

下面是两个例子：

```
sendmail.com.      IN TXT "v=spf1 ip4:209.246.26.40 ip4:63.211.143.38 ip4:
    209.246.26.36 ip4:209.246.26.12 ip4:209.246.26.18 ip4:209.246.26.10 ~all"
example.com.       IN TXT "v=spf1 MX PTR -all"
```

在第一行里，sendmail.com 这个域列出了其邮件服务器的 IP 地址。如果邮件自称来源于 sendmail.com，但又不是从 SPF 记录列表中的 IP 地址所指出的机器上发出的，那么这封邮件就是伪造的，应该被丢弃。第二行要求发电子邮件的机器为了被验证身份有效，都必须在 DNS 里有匹配的 MX 和 PTR 记录。

子句“v=spf1”指 SPF 协议的版本，目前只定义和实现了版本 1。有几种邮件传输代理（包括 **sendmail**、Postfix 和 **exim** 在内）都支持 SPF 处理。从 SPF 记录还可以引出许多其他花哨的功能，参

www.TopSage.com

考18.10.10节了解更完整的讨论。

TXT记录天生没有次序。如果使用多条记录把一段信息添加到DNS中，则根据**named**和UDP处理它们的时间，可能会被全部打乱。

15.7.12 IPv6资源记录

IPv6是IP协议的一个新版本。制订其规范的过程花费了10多年的时间，而且仍未完成[8]。IPv6的最初动机是感觉到需要更多的IP网络地址。不过，对这个问题的权宜之计——CIDR、NAT和对地址更严格的控制——已经如此成功，以至于没有实质性的要求，要像最初认识到的那样大规模向IPv6迁移。现在推动采用IPv6的是亚洲，那里分配的IPv4地址太少了。下一代移动电话可能就带IP地址，可能也有助于扩大采用IPv6的规模。

在DNS中支持IPv6的早期提议竭尽全力用A6和DNAME资源记录支持IPv6地址的共享所有权。虽然这些记录的类型让地址的重新分配更容易了，但它们太过复杂，以至于IETF因为喜欢简单得多的正向映射AAAA记录和反向映射ip6.arpa域，所以退回到了最初的计划上。我们在本书里将不再介绍A6记录、位串或者DNAME记录。如果您对此感到好奇，您可以参考本书以前的版本，或者参考RFC了解要它们怎样工作的细节。

15.7.13 IPv6正向记录

AAAA记录的格式为：

```
hostname [ttl] IN AAAA  ipaddr
```

例如：

```
anchor      IN  AAAA  3ffe:8050:201:9:a00:20ff:fe81:2b32
```

每个由冒号隔开的块都是4位十六进制数字，开头的0通常省略。两个相邻的冒号代表“为构成一个完整的IPv6地址，需要足够多的0填满128位”。一个地址至多可以包含一对这样的冒号。

15.7.14 IPv6反向记录

在IPv4里，反向映射在in-addr.arpa域里，正向映射在域名树的其他分支里（例如，在com或者edu下）。在IPv6里，和一条AAAA地址记录相对应的反向映射信息是在顶级域ip6.arpa[9]里的一条PTR记录。参考15.7.5节了解有关IPv4版PTR记录的讨论。

通过把冒号分隔的每段地址扩充为完整的4位十六进制数，然后再把那些位的次序颠倒过来，末尾加上ip6.int，这种“堆砌”格式就把一个AAAA地址记录反向过来了。例如，对应我们给anchor的AAAA记录，其PTR记录应为：

```
2.3.b.2.1.8.e.f.f.f.0.2.0.0.a.0.9.0.0.0.1.0.2.0.0.5.0.8.e.f.f.3.ip6.arpa PTR
anchor.cs.colorado.edu.
```

对于系统管理员来说，必须输入或者调试乃至阅读这样的格式肯定是不友好的。当然，在实际的

[8] IETF界的一位积极活动的成员Tony Li曾经这样描绘IPv6，“太少又太早”。

[9] 名字树上的IPv6反向分支最初叫做ip6.int。

www.TopSage.com

DNS 区文件里，$ORIGIN 语句会隐藏一些复杂性。

IPv6 还是新生事物，至少从部署的角度来看是如此。注册机构刚刚开始分配地址，随着经验的增长，这个过程将会越来越顺利。最近（2004 年）启动了一些广播 IPv6 地址的根名字服务器。默认该怎么做尚存疑问。例如，如果一个名字有 IPv4 地址而没有 IPv6 地址，那么在查它有没有 AAAA 记录时，应该说“没有这样的记录”吗？还是应该返回 IPv4 的 A 记录？

15.7.15 与安全有关的记录

与 DNSSEC 有关的资源记录（DNSKEY、DS、RRSIG 和 NSEC）构成了它们自己的一个主题。我们将在 15.15 节介绍 DNS 安全的时候讨论这些记录。这些记录从根本上说和大多数其他资源记录不一样，因为它们一般要用软件工具生成，而不是手工输入。

15.7.16 区文件中的命令

现在我们已经浏览了所有基本的资源记录，让我们看看可以嵌入到一个区文件中的命令，这些命令可以修改它们后面的记录。这些命令只是分析器的指令，它们有助于让区文件可读性更好，更易于维护。这些命令既不影响分析器对后续记录的解释，其自身也不会扩充成多个 DNS 记录。一旦区文件已经被读入和解释，这些命令就都不再是区数据的一部分了（至少，不再有它们原来的形式）。

这样的命令有 4 个：

$ORIGIN *domain-name*
$INCLUDE *filename [origin]*
$TTL *default-ttl*
$GENERATE *lots-of-args*

命令必须从第一列开始，并且整个命令要在一行。$ORIGIN 和$TTL 命令都由 RFC 所规定，所有的名字服务器都必须理解它们；$INCLUDE 和$GENERATE 最初仅 BIND 才有，但已经被一些别的 DNS 实现采纳了。

名字服务器读取区文件时，它将默认域（或“初始域”）加到尚不完整的任何域名之后。初始域最初设置为名字服务器配置文件中相应的 zone 语句所指定的域名。不过，您可以通过在区文件中使用$ORIGIN 指令手工设置初始域。

在要用完整域名的地方使用相对名节省了很多力气敲键盘，并让区文件更容易阅读。例如，对于一个划分了子网的 B 类站点的反向记录来说，就可以全在一个区文件中用$ORIGIN 语句为每个子网设置环境。就像这样一条语句：

```
$ORIGIN  243.138.128.in-addr.arpa
```

可以放在 243 子网的记录前面。

许多站点在区数据库文件中使用$INCLUDE 指令，把有开销的记录与数据记录分隔开，来把区文件分成若干个逻辑部分，或者把加密密钥保存在严格设定权限的文件中。$INCLUDE 指令的语法为：

```
$INCLUDE  filename[origin]
```

filename 指定文件在$INCLUDE 指令出现的地方读入数据库。如果 *filename* 不是绝对路径名，就按相对于名字服务器主目录来解释它。

$TTL 指令为其后面记录的存活时间（*ttl*）字段设置默认值。$TTL 指令应该在区的 SOA 记录之前。$TTL 值的默认单位是秒，但是您也可以用代表小时的 h、代表分钟的 m、代表天的 d 或者代表星

www.TopSage.com

期的 w 来修饰数字。例如，下面这几行：

```
$TTL  86400
$TTL  24h
$TTL  1d
```

都是把$TTL 设为一天。

$GENERATE 是一种相当新的结构，它提供了一种生成一系列类似记录的简单途径。它主要用来帮助生成 RFC2317 风格的无类 in-addr.arpa 映射（反向区文件的 CNAME 巧用记录），以防出现行政机构的边界与 IP 地址中的字节边界不匹配的情况。

$GENERATE 指令的格式为：

```
$GENERATE start-stop/[step] lhs type rhs [comment]
```

生成的行具有

```
lhs type rhs
```

的形式。

start 和 *stop* 字段指定数值循环的范围。在此期间的每一个值都生成一行。循环参数值用$字符嵌入到 *lhs*（左边界）和 *rhs*（右边界）中。如果还指定了一个 *step*，则循环按 *step* 的大小递增。*type* 是记录的类型。BIND 9 支持针对 CNAME、PTR、NS、DNAME、A 和 AAAA 的$GENERATE。参考 15.7.9 节中的例子。

15.7.17 粘合记录：区之间的链接

每个区都有自身的一套数据文件、名字服务器和客户机。但是各个区需要彼此连接起来以形成一个贯通的层次结构：cs.colorado.edu 是 colorado.edu 的一部分，在它们之间我们需要某种 DNS 链接关系。

因为 DNS 推荐都只从父域到子域，所以名字服务器没有必要知道 DNS 层次结构中在它之上的域（或更精确地说是区）的所有信息。不过，父域的服务器必须知道所有子域的名字服务器的 IP 地址。实际上，只有父区知道的名字服务器才能在响应外界查询时作为返回的推荐服务器。

就 DNS 而言，父区需要包含每个被授权区的 NS 记录。因为 NS 记录是用主机名而不是 IP 地址编写的，所以父服务器必须也有办法解析主机名，要么执行正常的 DNS 查询（如果这样不会创建一个循环的依赖关系），要么拥有适当的 A 记录副本。

满足这个要求可以有两种途径：直接包括必要的记录，或者使用存根区（stub zone）。

采用第一种方法，您可以简单地在父区中包括必要的 NS 和 A 记录。例如，colorado.edu 区文件可以包含这些记录：

```
; subdomain information

cs          IN  NS   ns.cs.colorado.edu.
            IN  NS   piper.cs.colorado.edu.
            IN  NS   ns.atrust.com.
ee          IN  NS   ns.ee.colorado.edu.
            IN  NS   ns.cs.colorado.edu.

; glue records

ns.cs       IN  A    128.138.243.151
piper.cs    IN  A    128.138.204.4
ns.ee       IN  A    128.138.200.1
```

www.TopSage.com

这些“外来”的 A 记录称为粘合记录，因为它们不真正属于该区。它们只是被复制到这儿，好把新域连接到 Internet 域名树。缺少或不正确的粘合记录将使您的部分域名空间变得不可访问，试图访问这个域名空间的用户将得到“host unknown（未知主机）”的错误信息。

一种常见的错误是，给不需要粘合记录的主机名包含了粘合记录。例如，前面例子中的 ns.atrust.com 可以用正常的 DNS 查询解析。最初没有必要用 A 记录，但是如果以后 ns.atrust.com 的地址改过了，那么这个记录会让人完全误解。原则是，只包括在当前域或其任何子域中的主机的 A 记录。BIND 的当前版本忽略不必要的粘合记录，并把它们的出现作为一个错误记录在日志中。

刚才描述的方案是连接区的标准方法，但是它要求子区与父区保持联系，并告诉父区对子区的名字服务器组所做的任何更改或添加。因为父区和子区常常由不同的站点运行，所以更新常常是一项很枯燥的手工劳动，要求跨越管理边界进行协调。在现实世界中，这种类型的配置常常就会过时，这是必然的。

保持链接的第二种方法是使用存根区。存根区本质上与从区是相同的，但是它仅包括父区的 NS 记录。自动更新存根区就可以避免在父区和子区间进行沟通。一个重要条件是，在父域的主服务器和从服务器上，存根区的配置必须相同。可能最简单的做法是，人为地与父域保持联系，并且在一年内至少验证几次它的配置（尤其是如果它在本地的话）。

您可以用 **dig** 命令查看父域当前正宣告的是哪些服务器。首先运行

```
$dig parent-domain ns
```

判断父域的名字服务器。选择一个并运行

```
$dig @name-server.parent-domain child-domain ns
```

来查看公共名字服务器列表。下面是一个实际的例子，**dig** 的一些废话被删掉了：

```
$ dig colorado.edu ns
;;     ...
;; ANSWER SECTION:
colorado.edu.          5h9m22s IN NS   ns1.westnet.net.
colorado.edu.          5h9m22s IN NS   boulder.colorado.edu.
colorado.edu.          5h9m22s IN NS   cujo.colorado.edu.

$ dig @boulder.colorado.edu cs.colorado.edu ns
;;; ANSWER SECTION:
cs.colorado.edu.       6H IN NS        cs.colorado.edu.
cs.colorado.edu.       6H IN NS        huizil.cs.colorado.edu.
cs.colorado.edu.       6H IN NS        anyns.pch.net.
cs.colorado.edu.       6H IN NS        pacifier.com.
```

从外界只能看到 cs.colorado.edu 域的 4 台名字服务器。在系里的一台机器上运行 **dig** 却能输出不同的结果：

```
;; ANSWER SECTION:
cs.colorado.edu.       2H IN NS        cs.colorado.edu.
cs.colorado.edu.       2H IN NS        moet.cs.colorado.edu.
cs.colorado.edu.       2H IN NS        piper.cs.colorado.edu.
cs.colorado.edu.       2H IN NS        anchor.cs.colorado.edu.
cs.colorado.edu.       2H IN NS        vulture.cs.colorado.edu.
```

注意，TTL 的值根据查询是从系内部还是系外来的而变化（2 小时和 6 小时的对比）。那是因为已经使用了 BIND 的 `view` 语句来定义数据的内部和外部视图（view），参考 15.11.13 节的内容。（两个值都取得短，几天到一周是个更好的选择。）

存根区特别有用的一种情况是，当您的内部编址使用 RFC1918 的专用地址空间，而且您需要

www.TopSage.com

保持同步 RFC1918 授权的时候。BIND 8 的软件发布在 **src/conf/recursive** 中有一个例子。

下面是存根区还值得一提的一些巧妙之处。

- 存根区不是区数据的权威副本，存根服务器不应该在区的 NS 记录中列出。
- 因为存根服务器没有在 NS 记录中列出，所以当区数据变化时，也不会自动通知它们。要更新存根服务器，您可以在主服务器的配置中添加 also-notify 子句，或者只是等待在由区的 SOA 记录中所指定的刷新时间间隔结束时更新存根区。timeout 选项在多数情况下都正常工作，尽管它可能潜在地会暂时破坏授权（详见 15.16.7 节）。
- 从理论上讲，如果名字服务器不能获得匹配的 A 记录，则它拥有区的 NS 记录的副本就没有用处。不过，名字服务器可以使用其配置文件中给定的主服务器 IP 地址来启动自身（zone 语句中的 masters 子句，参考 15.11.10 节的内容）。
- 为什么把您自己限制在 NS 记录上呢？为什么不是子域的二级服务器呢？这样也行。不过，如果父域的每个服务器也是一个子域的服务器，那么无法向下游服务器做任何推荐。父域服务器将为子域提供所有的 DNS 服务。可能这就是您希望的，也可能不是。

至此，我们已经介绍了一般适用于 DNS 系统及其数据库的大多数背景知识。在接下来的一节里，我们继续用 BIND 实现所特有的配置细节来介绍 DNS。

15.8 BIND 软件

BIND，即伯克利 Internet 名字域（Berkeley Internet Name Domain）系统，是 ISC（Internet Systems Consortium，Internet 系统联盟）的开放源代码软件包，它在 Linux、UNIX、Mac OS 和 Windows 上实现 DNS 协议并提供域名服务。

15.8.1 BIND 的版本

BIND 已经有 3 个主要版本：BIND 4、BIND 8 和 BIND 9。BIND 4 在 20 世纪 80 年代末期面世（大致对应于 RFC 1034 和 1035 的发布）。BIND 8 发布于 1997 年，而 BIND 9 则发布于 2000 年中期。没有 BIND 5、6 和 7 的版本，BIND 8 作为一个更新版本，其意义如此重大，以至于其作者觉得值得使用一个是老版本号两倍的版本号[10]。其实并非真的如此……BIND 8 是随 4.4BSD（Berkeley Software Distribution of UNIX，UNIX 的伯克利软件发布）一起发布的，因而所有的版本号都升为 8。**sendmail** 也跳过几个号码，同时达到版本 8。

BIND 8 融入了许多能够提高效率、稳定性和安全性的先进技术。BIND 9 甚至走得更远，它支持多处理器、线程安全的操作、真实性安全（公钥加密）、支持 IPv6、增量区传送和一些其他特性。一种新的数据结构（至少对于 BIND 是新的）红黑树（red-black tree）将区数据存储在内存中。BIND 9 经过了彻底的重新设计和重新实现。它将代码中针对操作系统的特定部分独立出来，因此能更容易地将 BIND 移植到非 UNIX 系统上。BIND 9 的内部机制有了显著不同，但是其配置步骤却保持不变。我们在本书里介绍 BIND 9。

15.8.2 确定现有的 BIND 版本

一个软件厂商提供文档，说明他们的系统所包含的一个外部软件包是哪个版本，这种情况似乎并

[10] 谁说营销和工程不可得兼？

www.TopSage.com

不常见，因此您可能不得不做些检测以准确地找出您正在使用的是什么软件。对于 BIND 而言，您有时可以通过 BIND 附带的一条命令 **dig**，暗地做一次查询来确定其版本号。除非有人已经决定通过更改配置文件中这则信息的内容来禁止提供版本信息，否则命令：

```
$dig @server version.bind  txt  chaos
```

将返回版本号。例如，对 isc.org 起作用的命令：

```
$ dig @ns-ext.isc.org version.bind txt chaos
version.bind.          0S CHAOS  TXT    "9.4.0a0"
```

就不能在 cs.colorado.edu 上得到结果：

```
$ dig @mroe.cs.colorado.edu version.bind txt chaos
version.bind.          0S CHAOS  TXT "wouldn't you like to know..."
```

有些站点配置 BIND 把自己的版本号给隐藏起来，理由是这样做能带来一定程度上的“通过隐蔽获得的安全性”。我们并不真正认同这种做法，但是它可能有助于避开一些用脚本的小黑客。参考 15.11.4 节，了解有关这个问题的详细讨论。

这次查询的结果里有“0S”。这个 0 是数字零，代表 TTL 的值。S 代表秒（second），但对于 TTL 来说默认的单位就是秒，所以通常并不打印出来。或许我们应该报告这个缺陷（bug）——乍看上去它就像代表操作系统的 OS。

IETF 正忙于把这个奇怪的 CHOAS 类区中的数据标准化，并将其总结好，这样一来，其他的 DNS 实现也可以采用这个约定。许多实现已经在这么做了，让变量名保持和版本无关，而其他实现则受到了包含“bind”的名字的困扰。因为 IETF 仍在争执到底这些变量的名字应该是什么，所以 ISC 在其当前的 DNS 软件发布中实现了所有备选的名字。例如，您也可以查询 CHAOS 类的变量 hostname.bind 或者更普通的 id.server。

通常您还可以通过检查**/var/log** 目录下的日志文件来确定 BIND 的版本。BIND 的服务器守护进程 **named** 在启动时将版本号记录到 syslog（设备“daemon”）。请参考第 10 章以获得有关 syslog 的更多信息。用 **grep** 查找类似下面这样的行：

```
Feb 23 00:25:13 senna named[433]: starting BIND 9.4.0a0 -c
    /var/named/named.ns-ext.conf
```

如果安装了 **named**，但是您的系统在引导时没有正常启动它，只要运行 **named -v** 即可，它将输出其版本号，然后退出。

表 15.7 显示了我们举例的系统中包括的 BIND 版本。在 Debian 和 Ubuntu 上，您可以选择安装 BIND 8 的 **bind** 软件包，也可以选择安装 BIND 9 的 **bind9** 软件包。低于 9.3.1 或者 8.4.6 的版本都有已知的安全问题，忘掉 BIND 4 吧。使用当前的发布版本最安全。

表 15.7　我们举例系统上采用的 BIND 版本

系　统	操作系统版本	BIND 版本
ISC	–	9.4.0b
RHEL	4.3	9.2.4-10，通过 rpm 安装
Fedora	FC5	9.3.2
SUSE	10.2	9.3.2
Debian	3.2b	8.4.6 或者 9.2.4
Ubuntu	6.06	8.4.6 或者 9.3.2

www.TopSage.com

大家网 TopSage.com

大多数 Linux 的发行商会把安全修正也补到比较老的版本上去。

15.8.3 BIND 的组成

BIND 系统由 3 部分组成：

- 称为 **named** 的守护进程，它回答查询；
- 库例程，它联系 DNS 分布式数据库的服务器，解析主机查询；
- DNS 的命令行接口，**nslookup**、**dig** 和 **host**。

以 DNS 的术语来说，像 **named** 这样的守护进程（或者运行它的机器）称为“名字服务器”，与其联系的客户端程序称为“解析器”。下面我们简要讨论每个部分的功能，而真正的 BIND 配置则放到后面的 15.11 节再讲述。

15.8.4 named: BIND 的名字服务器

named 回答有关主机名和 IP 地址的查询。如果 **named** 不知道某个查询的答案，它就会询问其他服务器并把它们的响应缓存下来。**named** 还执行“区传送”，在域内的服务器之间复制数据。（回忆一下，一个“区”是指一个域扣除它的各子域后的部分。名字服务器所处理的是区，但是“域[domain]”也经常用在实际意思是“区”的地方。）

名字服务器可以在几种不同的模式下运行。它们之间的差异落在了好几条线索上，因此最后的分类通常不太整齐。一个服务器可以在不同的区扮演不同的角色，这更容易让人产生混淆。表 15.8 列出了用来描述名字服务器的一些形容词。开头不缩进的是标题，标题下的缩进项是粗略的分类。

表 15.8 名字服务器的分类

服务器的类型	说 明
权威	一个区的正式代表
主服务器	区数据的首要服务器，从磁盘文件获得数据
从服务器	从主服务器复制得到它的数据（也称为次要服务器）
存根	和从服务器类似，但是仅仅复制名字服务器的数据（不复制主机数据）
分支	仅在域内可见[a]的服务器（也叫做“秘密服务器”）
非权威[b]	用缓存数据回答查询，不知道数据是否仍然有效
缓存	缓存以前的查询，通常没有本地区
转发器	代替许多客户机执行查询，创建一个大的缓存
递归	代替您查询，直到返回答案或者出错
非递归	如果不能回答查询，就将您引导至另一台服务器

a. 分支服务器对于知道其 IP 地址的任何人都可见。
b. 严格地讲，“非权威”是 DNS 查询响应的属性而不是服务器的属性。

这些分类是以名字服务器的数据源（授权、缓存、主、从）、保存数据的类型（存根）、查询途径（转发器）、发出回答的完整性（递归、非递归）以及服务器的可见性（分支）为基础的。下面几小节会详细介绍其中最重要的一些区别，其他区别则在本章的其他地方介绍。

15.8.5 权威与仅缓存服务器

主服务器（master）、从服务器（slave）和仅缓存（caching-only）服务器的差异在于两个特征：

www.TopSage.com

数据的来源，服务器对该域是否权威。

每个区都有一个主名字服务器。它在磁盘上保存本区数据的正式副本。系统管理员通过编辑主服务器的数据文件来更改区的数据。

从服务器通过“区传送”操作从主服务器获取它的数据。一个区可以有多台从名字服务器，但必须至少有一台。存根（stub）服务器是一种特殊类型的从服务器，仅装载主服务器上的 NS（名字服务器）记录。请参考 15.11.11 节了解为什么需要这种操作。对于同一台机器，最好既是您的区的主服务器，又是其他区的从服务器。这样的协作通常会带来良好的 DNS 邻里关系。请参考第 15.14.1 节以获得有关区传送的更多信息。

仅缓存名字服务器从一个启动文件加载一些根服务器的地址，并通过缓存由它解析的各个查询的回答来积累它的其余数据。仅缓存名字服务器自身没有数据，对于任何区都不是权威的，除外的可能是 localhost 区。

从名字服务器返回的权威性回答是“保证”准确的；而非权威性回答则可能是过期的。不过，非权威性回答有非常高的百分比是完全正确的。主服务器和从服务器对于它们自己的区来说是权威性的，但对于它们缓存的其他域的信息却不具权威性。老实说，如果系统管理员更改了主服务器数据却忘记传播这些改动（例如，没有更新数据的序列号），那么即使是权威性回答也可能不准确。

主服务器应该位于一台稳定的、用户不多、相对安全并且可能具有不间断供电的机器上。至少应该有两台从服务器，其中一台在本网点之外。本网点内的从服务器应该位于不同的网络以及不同的电源线路上。当域名服务停止时，所有正常的网络访问也都会停止。

尽管仅缓存服务器不是权威性的，但是它们可以减少用户感觉到的反应时间以及内部网络上的 DNS 流量。可考虑在每个子网上放置一台仅缓存服务器。在大多数网点上，桌面计算机通常访问一台仅缓存服务器来解析对 Internet 上主机的查询。

在 BIND 4 和 BIND 8 中，不要把一台名字服务器既作为您自己区的权威性服务器，又作为您用户的缓存服务器。每个 **named** 运行都要有一个在内存中的数据库，如果内存紧张，缓存数据与权威数据相互混合。BIND 9 已经解决了这个问题，所以消除了混合的情况。不过，安全和整体 DNS 的卫生仍然要求两种功能分开，一是向全世界提供权威数据服务的功能，二是向您的用户提供全世界数据服务的功能。

15.8.6 递归和非递归服务器

名字服务器要么是递归的，要么是非递归的。如果非递归服务器在以前事务的缓存中有对某一查询的回答，或它是被查询域的权威服务器，那么它就会提供正确的响应。否则，它并不返回真实的答案，而是推荐一个更有可能知道答案的其他域的权威性服务器。非递归服务器的客户机必须准备接受并对这些推荐服务器进行操作。

尽管非递归服务器可能看起来比较懒惰，但是它们往往有合理的理由不承担额外的工作。根服务器和顶级域服务器都是非递归的，但是在每秒 10 000 条的查询速度下，我们就可以理解它们这种走捷径的行为。

递归服务器仅返回真实的答案或错误消息。它自己跟踪这些推荐服务器，免除了客户端的这项任务。解析查询的基本过程在本质上是相同的，惟一的区别在于名字服务器会处理推荐服务器而不是将它们返回给客户端。

解析器库例程并不能理解返回结果是推荐服务器的情况，它们期望本地的名字服务器是递归的。

让名字服务器跟踪推荐服务器有一个副作用，即它的缓存获得了有关中间域的信息。在本地网络上，因为这样的缓存过程能让网络上任何主机的后续查询得益于名字服务器的前期工作，所以这经常是您希望的效果。另一方面，诸如 com 或 edu 这样的顶级域服务器不应该保存比它低几个域的主机所

www.TopSage.com

请求的信息。

BIND 的早期版本需要更改和重编译源代码以修改服务器的递归性质。这个选项后来转给一个命令行参数（**-r**），而如今则作为配置文件的一个参数。一个服务器甚至可以被配置为对自身的客户机是递归的而对外界的客户机是非递归的。

名字服务器按层次产生推荐服务器。例如，如果服务器不能提供主机 lair.cs.colorado.edu 的地址，它将推荐 cs.colorado.edu、colorado.edu、edu 或根域的服务器。推荐必须包括被推荐域的服务器的地址，因此选择并不是任意做出的，该服务器必须指向一个域，这个域必须已经知道这个服务器。

通常返回知道时间最久的域。如果不知道 lair 的地址，但是知道 cs.colorado.edu 的名字服务器，则返回这些名字服务器的地址。如果不知道 cs.colorado.edu，但是知道 colorado.edu，则返回 colorado.edu 的名字服务器的地址，以此类推。

名字服务器从列出根服务器的“线索”文件中预先加载其缓存的内容。即便是要“去询问根服务器”，也总是可以提出某种推荐的。

15.8.7 解析器库例程

客户机通过调用库例程的 **gethostbyname** 族函数来查询主机名的映射关系。最初实现的 **gethostbyname** 是在**/etc/hosts** 文件中查找域名。对于由 DNS 提供的主机映射关系来说，这些例程必须使用解析器的例程库，例程知道如何定位名字服务器并与之通信。解析器通常被集成到编译软件所要的标准库中。

大多数系统实现的 **gethostbyname** 可以从几个不同的来源提取信息：普通文件（例如**/etc/hosts**）、DNS，或许还有本地的管理性数据库系统（例如 NIS）。开关文件（switch file）能够对查找哪种来源以及查找顺序进行细致地管理控制。请参考 15.17 节和本书的 17.4.3 节以获得专门的信息。我们介绍特定发行版本上 DNS 系统的若干节深浅适宜地阐述了这个专题，内容适合于主机查询，具体从 15.17 节开始。

使用网络的应用程序一般编译时都链接了一个存根解析器库，它向一个本地的缓存名字服务器发送 DNS 查询请求。要支持 IPv6 就让事情复杂了不少，但是 BIND 9 提供了一个“轻量级的解析器库例程，而这些站点的解析器守护进程 **lwresd** 不需要支持 IPv6。就 DNS 而言，轻量级一词听上去像是个矛盾的说法，但是在这里，它是指解析器库和解析器守护进程之间使用的协议。如果在 **named** 的配置文件中包含 `lwres` 语句，那么名字服务器也会充当一个轻量级解析器。解析器守护进程现在不使用前面提到的名字服务的开关文件（switch file）：它只用 DNS 做域名解析。

15.8.8 DNS 的 shell 接口

BIND 软件发布包括 **dig**、**host** 和 **nslookup** 命令，它们都为执行 DNS 查询提供了一个命令行接口。它们作为调试辅助程序和从 DNS 提取信息的工具程序，有很大用处。尽管这些命令在功能上类似，但它们在设计上略有区别。请参考 15.16.6 节以获得更多信息。

15.9 设计 DNS 环境

许多因素会影响到为您特定的环境设计一个健壮而高效的 DNS 系统：您单位的规模、您是否在本地网路上使用 RFC 1918 中的私有 IP 地址、您是否使用微软的 active directory、您的内部网络是有

www.TopSage.com

路由的网络还是交换网络、以及您的防火墙和 DNS 服务器的位置关系是怎样的，都是这类因素。您可能会发现，把问题分成 3 个部分，会有所帮助：

- 管理名字空间的层次结构，子域、多级域名等；
- 向外界提供您站点的权威数据服务；
- 向您的用户提供名字查询服务。

15.9.1 名字空间管理

如果您的站点不大而且独立，那么除非您的上级出于某些非技术的原因，要求您使用子域，否则就既没必要也不值得。另一方面，在一个中等规模且有几个独立的系统管理小组的单位里，采用子域能够减少需要全单位协调的情况（按照地理或者部门来划分子域是最常见的做法）。一家大型单位几乎不会让全单位使用惟一的域名，所以就需要子域，或许还要分成若干级别。

创建子域要求在负责父域的系统管理员和负责子域的系统管理员之间进行沟通和合作。在授权并配置子域的时候，要保证记住，如果您想要添加、改变或者删除服务器，应该联系谁。如果您想要让单位之外的地方能访问到您的子域，那么还要保证您的防火墙不会阻挡对子域服务器的访问。

如果您使用子域来管理您的名字空间，可以每周从 **cron** 运行一次 **doc**（domain obscenity control，域名模糊控制）工具，确保授权保持同步，不会让您意外地弄出残缺的授权。本章将在 15.16 节介绍 **doc** 和其他几种对 DNS 健康有帮助的工具。

15.9.2 权威服务器

DNS 规范要求每个域至少有两台权威服务器。主服务器和从服务器是权威的，缓存服务器和存根服务器则不是权威的。在理想情况下，一个站点有多台权威服务器，每台都在一个独立的网络上，也有独立的电源线路。许多站点都有一台异地的权威服务器，经常由其 ISP 来托管。如果您的 ISP 没有提供这项服务，可以从一家 DNS 服务器提供商购买，或者和一家本地公司（理想的话，不是您的竞争对手）或大学进行交易。

几年以前，微软被抓住违反了名字服务器要分开放在不同网络的规矩。他们把自己的所有 3 台权威服务器都放在同一个子网上，把这个子网连接到 Internet 的那台路由器宕机以后，这些名字服务器就访问不到了。两个小时以后，随着缓存记录的作废，microsoft.com 及其所有别的域名都从 Internet 消失了。在根名字服务器上对微软有关的域名的查询量，从一般占总负载的 0.000 001%，增加到占总负载的 25%（每秒 10 000 次查询）。问题持续了两天。当尘埃落定之后，微软修好了这台路由器，并且将其 DNS 服务外包了出去！

权威服务器通过使用区传送来保持自己数据的同步。在从您主服务器到从服务器做区传送的时候，要使用 TSIG 密钥来验证身份和进行控制。参考 15.15.3 节了解有关 TSIG 的配置信息。

您可能想要您的权威服务器，在一定程度上根据查询人是谁来确定提供的查询响应。从您的网络之外来的查询可能会收到一个回答，而从您单位内部传来的同样查询会收到一个不同（更完整）的回答。这种配置叫做“split DNS[分离式 DNS]”，它是在区一级而不是服务器一级上实现的。

区的每个版本叫做“视图（view）”，它得名于在 BIND 配置文件里配置它的 `view` 语句。外面的人看到的是数据的一个视图，而内部的人看到的是另一个视图。这个特性通常用于把内部存在的计算机隐藏起来，不让人偷看到，再就是确保使用 RFC 1918 私有 IP 地址的机器不会漏到 Internet 上去。视图调试起来很困难，但是有了 BIND 的丰富日志功能，加上巧妙地利用 **dig** 命令，就能有所帮助，参考 15.16 节了解一些提示。

www.TopSage.com

15.9.3 缓存服务器

递归的缓存服务器回答本地用户对 Internet 上站点的查询。您网点里的每台计算机都应该能访问到一台本地的缓存服务器，是一台在同一子网上的服务器更好。

一定规模的单位应该考虑使用一种层次结构，其中有一两台机器作为“转发器”，本地子网的缓存服务器通过它们传自己的查询。转发器因而就形成了一个由多个子网共用的缓存。

根据您网点的规模，可以把转发器配置为独立机器，或者部署成一种分级结构。例如，子网服务器可能被转到一台部门服务器，接下来又被转到一台全网点的网关服务器。全站的转发器缓存了整个网点里用户请求查询的名字。这个配置把用于名字服务的对外带宽降至最低，而且能让本地的计算机共享一个大缓存。转发器将在介绍配置的 15.11.5 节里讨论。

如果一个缓存服务器死机了，那么对于是那台服务器的客户端的所有用户来说，网络实质上就停止工作了（您的电话就要开始响了）。用一个脚本启动您的缓存名字服务器，如果服务器死机了，这个脚本就会在几秒之后重新启动它们。下面是在一台运行着两个 **named** 实例的机器上一直保持连续运行的脚本。这个脚本接受一个参数，该参数标出了日志文件以及要用的名字服务器的配置文件：

```
#!/bin/sh

PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH
export PATH

trap "" 1
while :; do
  named -f -c /var/named/named.$1.conf >> /var/log/named.$1 2>&1 < /dev/null
  logger "named ($1) restart"
  sleep 15
done
exit
```

这个脚本每当要重新启动 **named** 时，先要等 15 秒（一个任意指定的值）再重启服务器，并且运行 **logger** 命令提交一个 syslog 日志项。

15.9.4 安全

DNS 的安全有自己完整一节（15.15 节）专门介绍。我们在这里就不再赘述了，但要提醒您记住，如果您使用了防火墙，就要保证您的 DNS 系统不会发出您的防火墙会封锁其应答的查询。这实质上意味着，您的 DNS 系统管理员应该持续同您的安全和网络系统管理员保持沟通。在默认情况下，DNS 的查询使用随便一个非特权的 UDP 端口（>1023），而应答是以同样端口为目的端口的 UDP 数据包。

15.9.5 总结

图 15.2 画出了前面几段内容推荐的设计。它显示出了一个两级的转发结构，这对于小型网点而言可谓绰绰有余。调整处理对外查询的服务器和处理向内查询的服务器之间的平衡，以便让任何一组服务器的负载都不至于太大。

还要注意使用异地的从服务器，我们高度推荐使用它。有多个办公地点的公司可以把一个地方定为主服务器，这个地点就成为相对于公司其他地方的“异地”。

www.TopSage.com

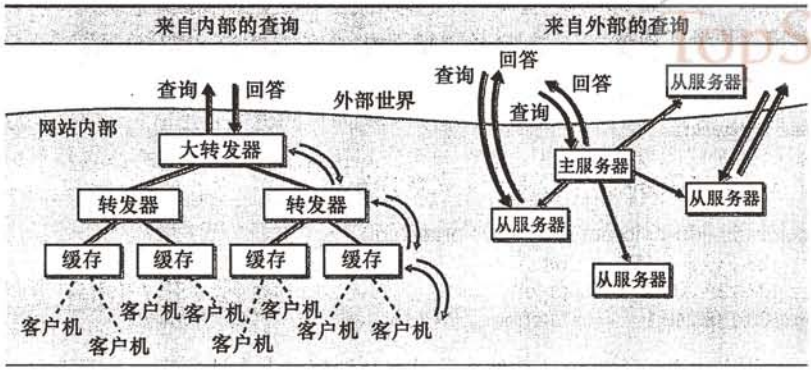


图 15.2 DNS 服务器的组织结构

15.9.6 DNS/BIND 管理工作分析

表 15.9 归纳了当您使用 BIND 和 DNS 时，必须做什么、为谁而做和做的频度。在“频度”列中所包含的“分发”字样意味着对每个子网或每种体系结构做一次，然后用类似 **rdist** 或 **rsync** 的工具程序将结果复制到相应的主机。请参考第 17 章以获得有关网络上分发文件的更多信息。

表 15.9 DNS 的安装和维护

任 务	对 象	频 度
获得域名	网点	一次
选择名字服务器	网点	一次或者多次
获得 BIND 软件	站点	一次，但是要得到当前版本
配置解析器	客户机	一次，并分发
配置有效的解析器	客户机	每个子网一次，并分发
配置服务开关文件	客户机	每种体系结构一次，并分发
在系统引导时启动 **named**	服务器	每个名字服务器一次
设置 **named** 的配置文件	服务器	每种类型的服务器一次
配置线索文件	服务器	一次[a]，并分发给服务器
配置区文件	主服务器	一次
更新区文件	主服务器	必要时
核查 log 文件	日志主机	每周至少一次
培训用户	所有主机	连续反复进行

a. 如果更改了根服务器，就必须重做。

15.10 BIND 客户机问题

由于网络上每台主机都必须是一个 BIND 客户机，所以我们开始详细讨论客户端的问题。

15.10.1 配置解析器

在网络上的每台主机都有一个叫做**/etc/resolv.conf** 的文件，它列出了主机应该查询的 DNS 服务器。

www.TopSage.com

如果您的主机从DHCP服务器获得IP地址和网络参数，那么应该自动设置**/etc/resolv.conf**文件。否则，您必须手工编辑它。其格式为：

```
search domainname ...
nameserver ipaddr
```

最多可列出3个名字服务器，以下是一个完整的例子：

```
search cs.colorado.edu colorado.edu ee.colorado.edu
nameserver 128.138.243.151    ; ns
nameserver 128.138.204.4      ; piper
nameserver 128.138.240.1      ; anchor
```

resolv.conf中从未定义过注释。该文件之所以在某种程度上支持注释，是因为它会忽略任何无法识别的内容。将注释信息放在`nameserver`行的最后是安全的，因为分析器仅查找一个IP地址，而忽略该行的剩余部分。因为`search`行能包含多个参数，所以加上注释信息会出问题。

`search`行列出了主机名不完整时要查询的域。例如，如果用户运行命令**ssh foo**，解析器就用搜索列表中的第一个域补充完整这个域名（在上述的**resolv.conf**中就是cs.colorado.edu）并查找foo.cs.colorado.edu。如果找不到这样的域名，解析器还将尝试foo.colorado.edu和foo.ee.colorado.edu。在一条`search`指令中可以指定的域的数量与解析器有关系，大多数解析器能够接受6到8个域。

resolv.conf中列出的服务器必须是递归的（因为解析器不理解推荐），并且每一个都应该有缓存。`nameserver`行中的服务器是依次访问的，只要第一个服务器能连续回答查询，那么其他服务器就被忽略。如果发生问题，那么查询就会超时，于是就尝试下一个名字服务器。按顺序尝试每个服务器，每个服务器最多试4次。每次操作失败将增加超时长度。

大多数解析器最多允许列出3个名字服务器。如果提供了更多的名字服务器，则多出的服务器会被悄悄地忽略掉。如果一台主机本身就是一个名字服务器，那应该在其自身的**resolv.conf**文件中首先列出它。如果没有列出名字服务器，那么就假定为localhost。

BIND的较早版本在**resolv.conf**中使用`domain`指令而不是`search`指令。它指定了一个用来追加给不完整名字的域名。我们推荐用`search`指令代替`domain`指令。这两个指令是互斥的，所以只能出现其中一个。如果您有一个较老版本的解析器，而且在**resolv.conf**文件中包括了这两条指令，那么使用的是列出的最后一条指令。

15.10.2 测试解析器

在大多数Linux发行版本上，只要在**/etc/resolv.conf**文件里有一个`nameserver`行，那么系统就会尝试在DNS里查询主机名。包括DNS在内，各种主机名信息来源的次序在**/etc/nsswitch.conf**中设定，请参见第15.17节。

配置好**/etc/resolv.conf**（并假定您的本地网络连接已启动并且运行正常）之后，应该可以通过名字而不是IP地址来访问其他机器。如果试图访问另一台本地机器，而命令却挂起了，请尝试通过其地址来访问该机器。如果能起作用，则问题出在DNS配置上。确认**/etc/resolv.conf**中名字服务器的IP地址正确无误，而且您指向的服务器允许从您的网络进行查询（参考15.11.4节）。从一台工作正常的机器上执行**dig**命令能回答这些问题。

15.10.3 对系统其余部分的影响

从静态主机表到DNS的变迁产生了引导和配置过程中一些潜在的依赖关系，对此需要加以保护。

www.TopSage.com

当主机引导时，如果在网络启动之前碰到启动文件里引用主机名，那么这些引用可能无法解析。启动文件中的命令尝试联络 DNS 将不会成功。由于解析器的健壮性，它们会对多个服务器进行多次尝试，每次尝试都会延长超时的时间长度。几分钟后，需要主机名的命令以失败告终。

为了解决这个问题，可在引导进程的早期阶段只使用纯粹的 IP 地址。或者，如果您的系统支持通过开关文件同时使用 DNS 和**/etc/hosts**，您就可以安装一个包含引导时需要的服务器地址的 **hosts** 文件。要确保在检查 DNS 之前检查 **hosts** 文件，这样就不必花时间等待 DNS 超时了。

15.11 配置 BIND 服务器

在本节中，我们假定您已经完成了行政上的琐碎事务。也就是说，假定您已经拥有一个域名（可能是一个子域），已经和父域的 DNS 管理员协调过，并在 in-addr.arpa 反向树中得到对您地址空间的授权。您已经选定了主名字服务器和几个从服务器，并且已经安装了 BIND。

15.11.1 硬件要求

BIND 十分消耗内存。它的数据库保留在内存中，所以随着 **named** 进程的运行，缓存会逐渐增大。BIND 9 的某些新特性也是 CPU 密集型任务，最突出的是 DNSSEC。为了有助于减轻这种负担，BIND 9 是多线程的，并且可以充分利用多处理器系统。BIND 9 还包含了能够控制 **named** 资源使用的配置选项。

判断一个名字服务器是否有足够的内存，最好的途径是运行一段时间，并观察 **named** 进程的大小。这个进程达到稳定的大小要花费一到两周的时间，此时老缓存记录的过期速度与插入新缓存记录的速度相同。一旦达到稳定，系统就不会发生交换（swapping），系统的调页（paging）率也应该在合理的范围内。

如果您的名字服务器运行在一台专门的机器上，那么最好让这台机器的内存是 **named** 运行两周之后它所消耗的内存数量的两倍。**top** 和 **vmstat** 命令可以显示出内存的使用情况，参考 25.3.3 节了解详情。

15.11.2 配置文件

named 的完整配置包括配置文件、根名字服务器的线索文件，对于主服务器来说，还有区数据文件，它包含每台主机的地址映射关系。**named** 的配置文件有它自己的格式，其他所有文件都是格式符合 DNS 规范的个别 DNS 数据记录的集合。我们将在接下来两节中讨论配置文件。DNS 数据记录的格式在 15.7.1 节介绍。

named 的配置文件是 **named.conf**，这个文件指定了这台主机的角色（主服务器、从服务器、存根服务器或者仅缓存服务器），它还指定了以何种方式获得它服务的每个区的数据副本。它还是指定选项的地方——这些选项既有和 **named** 整体运行有关的全局选项，也有针对服务器或者区的选项，它们只影响到了部分 DNS 流量。

named.conf 文件由一系列语句构成，每条语句以分号结束。各记号间以空白分隔，空白可以包括换行。大括号有时用于分组，但仅在特定的上下文关系中使用。这种格式很脆弱——漏掉一个分号就可能会造成严重破坏。

幸好 BIND 9 包含了两个方便的工具，它们可以检查配置文件的语法错误（**named-checkconf**）和区数据文件的语法错误（**named-checkzone**）。它们既查找错误，也会发现遗漏。例如，**named-checkzone**

www.TopSage.com

会告诉您是否忘记了包含一个$TTL指令。遗憾的是，它不能找出所有的错误。例如，漏掉了粘合(glue)记录（参见15.7.17节）的话是检查不出来的，因而给根域和gTLD服务器造成沉重负担。

注释可以出现在任何可以出现空白的地方。C、C++和shell风格的注释都是能够理解的：

```
/* This is a comment and can span lines. */
// Everything to the end of the line is a comment.
#  Everything to the end of the line is a comment.
```

每条语句以一个表明语句类型的关键字开始。除了options和logging之外，每种类型的语句都可以出现多次。还可以省略整条语句和部分语句，此时则调用缺失项的默认行为。表15.10列出了可以用的语句。

表15.10 在named.conf中使用的语句

语 句	功 能
include	插入一个文件（例如，仅named可读的可信密钥）
options	设置名字服务器的全局配置选项和默认值
acl	定义访问控制列表
key	定义验证信息
trusted-keys	使用预先设置的密钥
server	指定每个服务器所特有的选项
masters	定义一个主服务器列表，包括在存根区和从区里
logging	指定日志记录分类及其目的位置
zone	定义资源记录的一个区
controls	用ndc来定义用于控制名字服务器的渠道
view	定义域名空间的一个veiw（视图）
lwres	指定应该也是一个解析器的名字服务器

在说明这些语句和用它们来配置named的方式之前，我们需要描述在许多语句中使用的一种数据结构：地址匹配列表。地址匹配列表是IP地址的一种推广，它可以包括：

- IP地址，不是v4就是v6（例如199.165.145.4）；
- 用CIDR[11]网络掩码指定的IP网络（例如199.165/16）；
- 先前定义好的访问控制列表名；
- 加密的身份验证密钥；
- 用于取否的!字符。

地址匹配列表用作许多语句和选项的参数。例如：

```
{!1.2.3.13;1.2.3/24;};
{128.138/16; 198.11.16/24; 204.228.69/24; 127.0.0.1; };
```

这两个列表中的第一个排除了主机1.2.3.13，但是包含了1.2.3/24网络的其余部分。第二个列表定义了分配给科罗拉多大学的网络。括号和最后的分号实际上不是地址匹配列表的一部分，而是使用该

[11] CIDR网络掩码在12.4.4节介绍。

www.TopSage.com

列表的包含语句的一部分。

把 IP 地址或网络与匹配列表相比较时，要按顺序查找列表，直到找到相匹配的为止。“首先匹配”算法使这些项的顺序变得很重要。例如，如果将上述第一个地址匹配列表中的两项颠倒，则不会取得预期效果，因为 1.2.3.13 成功匹配 1.2.3/24，从而永远都不会碰到否定项。

现在就介绍这些语句。有的语句又简短又可爱，其他一些则几乎需要整整一章来深入描述它们。

15.11.3 include 语句

要划分或更好地组织大型配置，就可能要将配置的不同部分放置在不同的文件中。可以用 include 语句将补充文件引入 **named.conf** 文件：

```
include "path";
```

如果 *"path"* 是相对路径，则该路径就解释为相对于 directory 选项（随后介绍）中指定目录的路径。include 语句通常用来引入不是完全可读的加密密钥。不必禁止对整个 **named.conf** 文件的读取访问，而是将密钥放在仅由 **named** 可读的具有严格权限的文件中。这些文件以后可以包括到 **named.conf** 文件里。

15.11.4 options 语句

options 语句指定全局选项，对于特定区或服务器，某些选项以后可能会被覆盖。其一般格式为：

```
options {
    option;
    option;
    ...
};
```

如果在 **named.conf** 中没有 options 语句，则使用默认值。

BIND 8 大约有 50 个选项，BIND 9 的选项超过 100 个。要获得选项的完整清单，请查看 BIND 的文档[12]。我们侧重于讲解我们选出的选项，只讨论我们推荐使用的选项（我们也向 BIND 开发人员征求有关要包含哪些选项的意见，并采纳了他们的建议）。在每个选项旁边的中括号中列出了默认值。对于大多数站点而言，默认值就够了。列出选项的时候没有特别的次序。

```
directory "path"; [directory where the server was started]
```

directory 语句让 **named** 程序 **cd** 到指定的目录。不管相对路径名出现在 **named** 配置文件的什么位置，都将它们解释为相对该目录的相对路径。*path* 应该是一个绝对路径。任何输出文件（调试、统计等）也写入这个目录。

我们推荐将所有与 BIND 相关的配置文件（除了 **named.conf** 和 **resolv.conf**）放在**/var** 之下的子目录中（或者您保存其他程序的配置文件的地方）。我们使用**/var/named**。

```
version "string";       [real version number of the server]
hostname "string";      [real hostname of the server]
server-id "string";     [none]
```

version（版本）字符串标识在这台服务器上正在运行的名字服务器软件的版本；hostname（主机

[12] 在 Linux 发行版本 doc/misc/options 目录下的文件里有配置语言的语法以及所有的选项。

www.TopSage.com

名）字符串标识这台服务器自己，和 server-id（服务器 ID）字符串的作用一样。这两个选项能够让您不用真实值。每个选项把数据放入 CHAO 类的 TXT 记录里，好奇的旁观者会到那里去搜索它们。

对于隐藏名字服务器软件版本号的问题来说，有两类想法。有些网点认为，如果黑客们知道该网点运行的版本，那么网点的服务器就更容易受到攻击。其他网点则认为隐藏版本信息达不到预期目的，因为无论如何黑客们总在不断碰运气，而且在该软件的所有版本中都会发现最新的缺陷。

我们建议您不要重新设置版本的字符串。因为那样可以很方便地查询您的名字服务器并查出它们运行的版本（例如，如果您希望知道您的厂商是否提供了一个当前版本，或者如果您需要验证您是否已经将所有服务器升级为最新版本）。

主机名和服务器 ID 这两个参数是最近才加进来的，其动机是使用任播路由来复制根和 gTLD 服务器的实例。例如，在全世界散布着 20 个 f.root-servers.net 的运行实例，对于一个尝试调试问题的用户或者系统管理员来说，您对于到底是 20 个实例中的哪一个回答（或者无法回答）您的名字查询一无所知。查询 CHAOS 类中的 hostname.bind 会告诉您是 F 根服务器的哪一个实例在回答查询（只要在此期间，路由没有发生变化）。“server-id”是 hostname.bind 正确的名字，因为不是所有的名字服务器都运行 BIND。

```
notify yes | no;                    [yes]
also-notify servers_ipaddrs;        [empty]
```

如果 notify 设为 yes，并且该 **named** 是一个或多个区的主服务器，那么每当区数据库有变化时，**named** 将自动通知相应区的从服务器。然后，从服务器可以和主服务器协商来更新区数据的副本。notify 选项既可以用作全局选项，也可以用作针对区的特定选项。它使得您做出更改后，区文件收敛得更快。

named 通常通过查看某个区的 NS 记录来找出哪些机器是该区的从服务器。如果指定了 also-notify，那么还会通知其他一些没有用 NS 记录公告的服务器。当您的网点有内部服务器时，这个方法有时是必要的。不要用 also-notify 选项来通知存根服务器，存根服务器只对区的 NS 记录感兴趣，并可以等待常规的周期更新。请参考 15.11.11 节以获得有关存根区的更多信息。

```
recursion yes | no;                          [yes]
allow-recursion { address_match_list };      [all hosts]
```

如 15.8.6 节所述，recursion 选项指定 **named** 是否代替客户机查询其他名字服务器。通常不把名字服务器设置成关闭递归查询。不过，您可能希望对您自身的客户机允许递归，但对外来查询禁止递归。

递归可以通过 allow-recursion 选项和一个地址列表进行更细粒度的控制，该列表包括您希望执行递归查询的主机和网络。

```
transfer-format one-answer | many-answers; [many-answers]
```

这个选项影响将 DNS 数据记录从主服务器复制到从服务器的方式。在过去，数据记录的实际传输通常是每次传输一条记录，这是一种迟钝而效率低下的方式。BIND 8.1 增加了将多条记录组合成一个包的选项（many-answers），这在 BIND 9 中是默认的。如果是混合环境，则可以在个别 server 语句中指定传输格式，以覆盖全局选项。服务器的混合环境会决定是否在全局选择 many-answers，并对特定服务器覆盖该选项，或者反过来。

```
transfers-in number;                [10]
transfers-out number;               [10 (V9 only)]
transfers-per-ns number;            [2]
transfer-source IP-address;         [system dependent]
```

www.TopSage.com

一个大型站点——服务一个非常大的区的站点（例如 com，如今已经超过 2G 字节），或者服务成千上万个区的站点——或许需要调整这些区传送选项中的某几个。

transfers-in 和 transfers-out 选项限制可能同时发生的入站或出站的区传送数目[13]。transfers-per-ns 选项设置从同一个远程服务器可以并发运行的入站区传送的最大数目。大型站点可能需要增加 transfers-in 或 transfers-out 的值，注意不要用光了 **named** 进程的文件描述符。transfers-per-ns 可能不应该改动，它控制着主服务器乐意奉献给我们的资源数量。仅当所有远程主服务器都乐于处理多于两个的同时区传送时，才增加该选项。以每个服务器为基础，用 server 语句的 transfers 子句来进行更改是微调收敛从区数据的一种较好方法。

transfer-source 选项让您指定希望用于入站传输的接口的 IP 地址。它必须与主服务器上 allow-transfer 语句中指定的地址匹配。

和您对任何参数做较大改动时一样，在更改这些阈值中的一个之后，您应该仔细观察效果，保证机器不会崩溃。日志文件是您的好帮手。

```
files number;     [unlimited]
datasize number; [OS default value, system dependent]
```

files 选项设置服务器运行时允许同时打开的最大文件数。默认值尽可能贴近内核能支持的打开文件数（要改变内核支持的打开文件数，可以设置**/proc/sys/fs/file-max** 的值。参见 28.4 节）。

datasize 选项设定允许服务器使用的最大数据内存量。这个选项的单位是字节，但是 **named** 能理解 G 代表千兆字节、M 代表兆字节等。这个选项不应该用来试图控制缓存的大小，它是一个硬性的限制，根据在代码中碰到这个限制是在什么地方，可以导致名字服务器崩溃或者挂起。通常用它增加操作系统默认的单个进程的最大内存量限制。要控制缓存的大小，可以代之以使用 max-cache-size 和 recursive-clients 选项。

```
max-cache-size number;    [unlimited]
recursive-clients number; [1000]
```

max-cache-size 选项限制服务器可以用于缓存查询答案的内存量。如果缓存增长得太大，就要在记录的 TTL 到期之前先将它们从缓存中删除，从而确保内存的用量保持在稳定的限度内。

recursive-clients 选项限制服务器处理的并发递归查询的数量。每个递归查询都要在服务器上消耗一定量的内存，内存有限的服务器需要把这个限制降到比默认值还要低。

```
query-source address ip_addr port ip_port; [random]
listen-on port ip_port address_match_list; [53 any]
```

query+source 选项指定 **named** 用来查询其他名字服务器的网络接口和协议端口。listen-on 选项指定 **named** 监听查询请求的网络接口和协议端口。这两个选项默认取 **named** 的标准值：监听所有网络接口上的 53 端口，使用一个随机的、高端口号的 UDP 端口和任何网络接口做查询。

如果您的网点有防火墙，您就可以使用 query-source 选项，为外部 DNS 查询提供一个专门的、可识别的样板。有些网点设置 **named** 使用端口 53 作为源端口（也是被监听的端口），这样一来，防火墙可以把出站的 DNS 流量识别为源自于您的某一台名字服务器、可信任的包。不过，这个习惯也让过滤器很难分辨进出的 DNS 流量。您可以使用 53 之外的另一个端口，只要防火墙知道哪个端口是哪个就行了。

不要把源端口设为 0——那是一个非法的端口，**named** 会把查询请求当作是错误写到日志里去，

[13] BIND 代码内 transfers-in 参数的硬性限制为 20。

www.TopSage.com

不会回答它。有家大型ISP的系统管理员喜欢用端口0，他用query-source子句就让他们的许多名字服务器都不干活了。奇怪的是他们的客户居然没有发觉，也没有投诉。

```
avoid-v4-udp-ports { port_list }; [empty]
avoid-v6-udp-ports { port_list }; [empty]
```

如果没有使用这个query-source选项，BIND 9一般会用任何对外的网络接口，并且从非特权端口范围内（>1023）随机地分配一个源端口。如果网点的防火墙封锁了这个范围内的某些端口，那么就造成了一个问题。例如，有些防火墙不正确地认为1024端口属于特权端口范围，或者不正确地封锁了外界的2049端口（SunRPC）。当名字服务器发出一个查询请求，而且使用了一个被封的端口作为查询请求的源端口，防火墙就会封锁应答，名字服务器最终将放弃等待应答，再次发出查询请求。这虽说不是个致命问题，但是会让用户又气又恼。

为了避免出现这个问题，可以使用avoid-v4-udp-ports选项，让BIND避开被封的端口。被防火墙封锁的任何高端口号的UDP端口都应该包括在这个列表里[14]。如果您为了对付某些有威胁的攻击而更新了防火墙，那么要确保也要更新这里的端口列表。

```
forwarders { in_addr; in_addr; … }; [empty list]
forward only | first;              [first]
```

不用让每台名字服务器自己执行外部查询，您可以将一台或多台服务器指定为转发器（forwarder）。一般服务器可以核查其缓存的记录以及它具有权威性的记录，如果它没有找到要查找的答案，就会向一台转发器主机发送查询。这样一来，转发器就创建出整个网点都可以利用的缓存。这个指定是不明确的——转发器的配置文件里没有说“嗨，你是一个转发器”。

forwarders选项列出了要用来作为转发器的服务器的IP地址。它们被依次查询。转发器的使用绕过了从根服务器开始、沿着推荐关系链检索的正常的DNS过程。注意不要出现转发循环。

forward only服务器缓存数值并查询转发器，但是从不查询其他任何机器。如果转发器不响应，查询便失败。forward first服务器侧重于与转发器打交道，但是如果需要，还能直接处理查询。

既然forwarders选项没有默认值，所以除非特意配置它，否则不会进行转发。您可以全局地或在个别zone语句中打开转发功能。

```
allow-query { address_match_list };     [all hosts]
allow-transfer { address_match_list }; [all hosts]
blackhole { address_match_list };       [empty]
```

这几个选项让您指定哪些主机（或网络）可以查询您的名字服务器以及请求区数据的块传输。blackhole地址列表标识出您从不希望与之进行通信的服务器；**named**将不接受来自这些服务器的查询，也不会向它们询问答案。

```
sortlist { address_match_list }; [should die, don't use]
```

我们提这个sortlist选项只是要提醒您远离它。它最初的目的是帮助不能对记录集合进行正确排序的原始解析器。它让您指定多个答复返回的顺序，它的工作方式违背了目前BIND的内部机制。

其他与排序有关的BIND选项有rrset-order语句，它指定是按固定循环次序（轮流）还是按随机次序返回多条回答，还有topology语句，它尝试再次评估BIND的系统，选择要查询的远程服

[14] 有些防火墙是有状态的，可能聪明到能认出DNS应答，把它和1秒前相应的查询配成对。这样的防火墙不需要这个选项的帮助。

www.TopSage.com

务器。在大多数情况下也没有必要使用这些语句。

```
lame-ttl number;        [600 (10 minutes)]
max-ncache-ttl number; [10800 (3 hours)]
max-cache-ttl number;  [604800 (1 week)]
```

TTL 选项给存活时间（time-to-live）值设限，这些值控制着如何缓存肯定、否定和无响应的查询结果。所有传入的资源记录来的时候都有它们自己的 TTL，但是这些选项让本地服务器自己设限，为的是减少缓存数据所占用的内存。max-的选项是 TTL 值的上限，一定不要增加它们。DNS 的规范要求否定回答的 TTL（例如，“no such domain”）要比真实数据的 TTL 短。

lam-ttl 指定了缓存一个残缺服务器指示的秒数。最大值为 1800 秒，即 30 分钟。大多数残缺服务器（参见 15.16.7 节）会无限期地保持残缺，因为它们的所有者不知道它们没有和自己的父区同步。把这个参数设为最大值应该比较好。max-ncache-ttl 给否定缓存的 TTL 设限。它默认为 3 小时，最大值为一星期。max-cache-ttl 给一个查询的肯定回答缓存的 TTL 设限。

```
enable-dnssec yes | no; [no]
```

这个选项配置对 DNSSEC 的支持。目前的默认值为 no，但是在一两年内有希望切换到 yes。

```
edns-udp-size number; [4096]
```

UDP 包古老的默认长度为 512 字节，Internet 上所有的机器都要求能把被分片的、长度小于等于 512 字节的 UDP 包重新拼装起来。尽管这一限制在 20 世纪 80 年代中期有意义，但是后来就不再有意义了。现在，这个限制严重地影响了像名字服务器这样，要用 UDP 查询和响应的程序。

BIND 9 把默认的 UDP 包长设为 4096 字节（允许的最大值），使之能容纳 DNSSEC、IPv6 和国际化域名这样的新特性。不过，有些（不完善）的防火墙不允许 UDP 包长超过 512 字节，如果您在这样的一个防火墙后面，那么就需要这个选项，作为一种变通手段，直到您修正了防火墙为止。其值的合法范围介于 512~4096 字节之间。

```
ixfr-from-differences yes | no; [no]
```

以前，只有采用动态更新（在 15.12.2 节介绍）的区才支持增量区传送（IXFR）。有了这个选项，手工维护的区（例如，用一个文本编辑器）现在也能利用增量区传送了。IXFR 以多花一点儿 CPU 和内存、对区记录排序、计算和应用数据的差异为代价，节省了网络带宽。这个选项使用了和动态更新一样的健壮的日志机制。

```
root-delegation-only exclude { namelist };[empty]
zone "name" { type delegation-only; };
```

VeriSign 的 Site Finder 工具试图用一种通配的 DNS 记录，把每个人向其服务器敲错的域名重定向到别处去。人们已经通过政治手段，力求解决这种形式的“劫持 hijacking 攻击”（译者注：即网站被攻击后每次访问都自动跳转至另一个页面或者网站），但与此同时，BIND 9 实现了一种 delegation-only 类型的区、一个 delegation-only 区的选项，和全局配置 root-delegation-only 来作为解决这个问题的一个权宜之计。这两个选项限制根和 TLD 服务器只提供授权信息（名字服务器记录和名字服务器的地址），并且禁止它们提供主机数据。在启用了两者的功能之后，您的本地缓存服务器在回答敲错的请求时，就会返回“no such domain”。

之所以加入 exclude 子句，是因为不是所有的 TLD 都是 delegation-only 的区。例如，museum、us 和 de 都不是，但 com、net 和 org 应该是。下面从 cs.colorado.edu 域的区文件中截取的片段演示了使用新类型的区使得 Site Finder 失效的例子。参考 15.6 节了解有关 Site Finder 的更多背景资料。

www.TopSage.com

```
// to deal with VeriSign Site Finder crap
zone "com" { type delegation-only; };
zone "net" { type delegation-only; };
```

15.11.5 acl 语句

访问控制列表就是一个有名字的地址匹配列表：

```
acl acl_name {
    address_match_list
};
```

除了 also-notify 语句之外，您可以在需要调用地址匹配列表的任何位置使用访问控制列表。

acl 必须是 **named.conf** 中的顶级语句，所以不要试着暗中在其他选项的声明中包括它。**named.conf** 只读一遍，所以访问控制列表必须在使用之前定义。有 4 个列表是预先定义的：

- any——所有主机；
- localnets——本地网络上所有主机；
- localhost——机器本身；
- none——没有任何主机。

localnets 列表包括了主机直连的所有网络。换句话说，它是通过机器上的网络地址（一个或者多个）对网络掩码求模得出的。

15.11.6 key 语句

key 语句定义了用于某个特定服务器身份验证的有名字的加密密钥。有关 BIND 对加密身份验证的背景知识将在 15.15 节里介绍。这里我们只是简要地说明这个过程的机制。

若要创建一个密钥，您要指定想用的加密算法和一个用 base-64 编码串表示的“共享秘密”（也就是说，口令）：

```
key key-id {
    algorithm string;
    secret string;
};
```

和访问控制列表一样，*key_id* 必须在首次使用之前先在 **named.conf** 文件中定义。若要将该密钥与某个特定服务器关联起来，只需在该服务器的 server 语句的 keys 子句中包括 *key_id*。这个密钥既用来验证来自那个服务器的请求，又用来对那些请求的响应作签名。

15.11.7 trusted-keys 语句

trusted-keys 语句用于实现 RFC2535 中规定的 DNSSEC 安全机制。该语句的每一项都是一个 5 元组，标识出与该域的名字服务器进行安全通信所需的域名、标志、协议、算法和密钥。格式为：

```
trusted-keys {
    domain flags protocol algorithm key;
    domain flags protocol algorithm key;
    ...
}
```

www.TopSage.com

每一行代表用于某个特定域的信任密钥。*flags*、*protocol* 和 *algorithm* 是非负的整数。*key* 是一个 base-64 编码串。

当对某个区进行了签名但没有对其父区进行签名，因而无法肯定从 DNS 获得的该区的公钥是否合适时，就可使用 trusted-keys 结构。用一条 trusted-keys 语句（使用非常规方法）输入一个信任密钥可以保证您确实拥有那个有疑问的域的正确密钥。

在 15.15.4 节将会更详细地介绍 DNSSEC。

15.11.8 server 语句

named 有可能与许多服务器进行通信，但其中并非所有的服务器都运行着 BIND 的最新版本，甚至在表面上看，也并非所有的服务器都是健全的。server 语句告诉 **named** 有关其远程对等服务器的特征。

```
server ip_addr {
    bogus yes | no;                              [no]
    provide-ixfr yes | no;                       [yes]
    request-ixfr yes | no;                       [yes]
    edns yes | no;                               [yes]
    transfers number;                            [2]
    transfer-format one-answer | many-answers;   [many-answers]
    keys { key-id; key-id; ... };
    transfer-source ip-address [port]
    transfer-source-v6 ipv6-address [port]
};
```

您可以使用 server 语句来覆盖与服务器相关的配置选项的值。只列出您不希望具有默认行为的服务器。

如果将服务器标记为 bogus，**named** 就不会发送任何查询。这个指令应该一般是为实际上的伪（bogus）服务器保留的。bogus 不同于全局选项 blackhole（黑洞），因为它只是不允许向外的查询。与此相对照，blackhole 选项则完全排除了与所列服务器的一切形式的通信。

当 BIND 9 的名字服务器是作为一个区的主服务器运行时，如果把 provide-ixfr 设为 yes，那么它将执行增量区传送。类似地，当一个名字服务器是作为从服务器运行时，如果把 request-ixfr 设为 yes，那么它将从主服务器请求增量区传送。

edns 子句决定了本地服务器在联络远程服务器的时候，是否尝试使用扩展的 DNS 协议（EDNS）。BIND 中的许多新特性（例如，IPv6 和 DNSSEC）产生的包都会比 512 字节大，因此要求使用 EDNS 协议来协商使用更大的 UDP 包。

从服务器通过在数据变化时接收一次区传送来和它们的主服务器保持同步（参考 15.14.1 节了解有关区传送的更多知识）。transfers 子句限制了从远程服务器来的向内区传送的并发数量。它是 transfers-in 针对服务器的特定版本，但由于它仅适用于一个服务器，因此看起来就像是各个服务器对 transfers-per-ns 选项的覆盖。为了保持与 BIND 8 的兼容性，使用了不同的名称。

transfer-format 子句是前面 15.11.4 节讨论的选项针对服务器的特定形式。如果要同时与 BIND 8/9 和 BIND 4 服务器通信，或者如果在网络上有旧的微软名字服务器，那么就可以使用 transfer-format。除了一次回答（one-answer）格式之外，Windows NT 不能处理任何别的格式，Windows 2000 及以后的版本能处理两种格式的任何一种，尽管其中早一些的版本不能处理超过 16K 大的 DNS 消息。

keys 子句识别先前已经在 key 语句中定义的密钥 ID，用于 TSIG 事务签名（参见 15.15.3 节）。发送到远程服务器的任何请求都用该密钥签名。远程服务器产生的请求不要求进行签名，但如果签了，

www.TopSage.com

签名就要得到验证。

transfer-source 子句给出了网络接口的 IPv4 或者 IPv6 地址（还可以给出端口），应该将它们作为区传送请求的源地址（端口）。这个子句只有当系统有多个网络接口，而远程服务器已经在其 allow-transfer 子句中指定了一个专门的 IP 地址时才需要用。

15.11.9 masters 语句

masteres 语句让您给一个或者多个主服务器起名字，方法是用它们的 IP 地址和密钥。接下来您就可以把这个名字用在 zone 语句的 masters 子句里，用不着重复 IP 地址和密钥。怎样才能有多个 masters？参考 15.11.4 节。

当多个从服务器或者存根服务器从相同的远程服务器取得自己的数据时，这个 masters 工具就能帮上忙。如果那个远程主机的地址或者密钥变了，您可以更新这个 masters 语句，把它们引入，不用去改许多不同的 zone 语句。

它的语法为：

```
masters name { ip_addr [port ip_port] [key key] ; ... } ;
```

15.11.10 logging 语句

named 目前享有“地球上可配置性最好的日志系统”的声誉。系统日志把日志消息的优先次序安排交给程序员，而把对这些消息的处理交给系统管理员。但是对于给定的优先次序，系统管理员不能说“我关心这条消息，但不关心那条消息”。BIND 8 增加了一些类别，按类型将日志消息分类，还增加了一些渠道，拓宽了消息处理方式的选择。类别由程序员决定，渠道由系统管理员决定。

因为日志问题有些离题（特别是给出必需的解释），所以我们留到本章后面介绍调试的章节再讨论，也就是 15.16.1 节。

15.11.11 zone 语句

zone 语句是 **named.conf** 文件的核心部分。它们告诉 **named** 它具有权威性的区，并为管理每个区设置适当的选项。zone 语句还用于预先装载根服务器线索——根服务器的名称和地址，作为 DNS 查询过程的起点。

zone 语句的准确格式根据 **named** 在该区中所起作用的不同而不相同。具体来说，区的类型可能有：master（主）、slave（从）、stub（存根）和 delegation-only（仅授权）。我们在 15.11.4 节介绍过 delegation-only。其他类型在下面简要说明。

前面介绍的许多全局选项都可以成为 zone 语句的一部分并覆盖先前定义的值。在这里，除了会提到某些被频繁用到的特定选项之外，我们不再重复介绍这些选项。

一、配置一个区的主服务器

这里是 **named** 作为一个区的主服务器所需要的格式：

```
zone "domain_name" {
    type master;
    file "path";
};
```

在一个区的规范定义中，*domain_name* 一定要用双引号括起来。

区的数据保存在磁盘上的一个文件里，常人可以阅读（并编辑）这个文件。因为没有默认的文件

www.TopSage.com

名，所以您必须在声明一个 master 区时提供一条 file 语句。一个区文件只是 DNS 资源记录的集合，这些资源记录的格式在 15.7 节介绍过。

经常还会指定一些其他的特定于服务器的属性。例如：

```
allow-query { address_match_list };    [any]
allow-transfer { address_match_list }; [any]
allow-update { address_match_list };   [none]
zone-statistics yes | no               [no]
```

访问控制选项并不必要，但用上它们是个不错的主意。如果这个区使用动态更新，那么就会出现带有一个地址匹配列表的 allow-update 子句，这个列表限制可以发生更新的主机。动态更新仅适用于 master 区，allow-update 子句不能用于 slave 区。要确保这个子句包括的只是本地 DHCP 服务器，而不是整个 Internet[15]。

zone-statistics 选项能够让 **named** 保留查询/响应的统计信息，比如响应是推荐的其他域名服务器（而不是最终结果）的数量和百分比、响应是要求递归查询的数量和百分比，或者响应是导致出错结果的数量和百分比。参考 15.16.5 节中给出的例子。

有了所有这些针对区的特定选项（以及我们未作介绍的其他 30 多个选项），配置工作开始变得复杂起来。不过，一个主区的声明只包含区文件的路径名而不包含其他的任何内容，这也是完全合理的。在 BIND 4 中，那正是您所能指定的全部内容。这里是取自 BIND 文档中的一个例子，我们稍微做了修改：

```
zone "example.com" {
    type master;
    file "forward/example.com";
    allow-query { any; };
    allow-transfer { my-slaves; };
}
```

这里的 my-slaves 是先前定义的访问控制列表。

二、配置一个区的从服务器

从服务器的 zone 语句和主服务器的非常相似：

```
zone "domain_name" {
    type slave | stub;
    file "path";
    masters { ip_addr [port ip_port] [key keyname]; ... };   [no default]
    allow-query { address_match_list };                      [any]
    allow-transfer { address_match_list };                   [any]
};
```

正常情况下，从服务器会保留区数据库的一个完整副本。不过，如果服务器的类型设置为 stub，而不是 slave，那么只传送 NS（名字服务器）记录。存根区能让父区的 **named** 自动检测哪些机器为它们授权的子区提供 DNS 服务，以防出现子区的管理员不能尽责地把发生的变动通知给父区这样的情况。父区需要这一信息来做出正确的推荐或递归查询。

file 语句指定一个存储数据库副本的本地文件。服务器每次取回区的新副本时，它就把数据保存在这个文件中。如果服务器崩溃并重新启动，那么可以从本地磁盘重新装载这个文件，而不必通过网络传送。

您不应该编辑这个缓存文件，因为它是由 **named** 维护的。不过，如果您怀疑您在主服务器数据文件中发生了一个错误，那么可能会有兴趣看看这个文件。从服务器的磁盘文件向您展示了 **named** 是如何解释原来的区数据的，相对名和原指令全都进行了扩展。如果您在数据文件中看到类似以下之一的

[15] 您还需要在防火墙上进行流入过滤，请参考 20.12.1 节。还是用 TSIG 作身份验证更好。

www.TopSage.com

域名：

```
128.138.243.151.cs.colorado.edu.
anchor.cs.colorado.edu.cs.colorado.edu.
```

可以非常肯定，您在某处忘记了结尾的一个点（.）。

masters 语句列出了一台或多台机器的 IP 地址，可以从这些 IP 地址得到区数据库。它还可以包含在上一条 masters 语句中定义的一个主服务器列表的名字。

我们曾经说过，只有一台机器可以作为一个区的主服务器，那么为什么还可能列出多个地址呢？原因有两个：

首先，主服务器可能具有多个网络接口，因此具有多个 IP 地址。有可能一个接口变成了不可达（因为网络或路由问题），而其他的接口仍然可以访问到。因此，最好列出主服务器在拓扑结构上有区别的所有地址。

其次，**named** 实际上并不关心区数据的来源。它可以像从主服务器那样轻松地由从服务器上取出数据库。您可以运用这个特性让一个连接良好的从服务器作为一种备份主服务器，因为 **named** 会按顺序尝试所有的 IP 地址，直到找到一个能工作的服务器为止。理论上讲，您还可以创建服务器的层次结构，一台主服务器服务于几个二级的服务器，而二级服务器接下来又服务于许多三级服务器。

三、设置根服务器的线索

zone 语句的另一种形式把 **named** 指向一个文件，**named** 可以通过该文件用根名字服务器的名称和地址预先填充（预装载）其缓存。

```
zone "." {
    type hint;
    file "path";
};
```

“hint（线索）”是一个列出根域（“.”）服务器的 DNS 记录的集合。需要用它们为 **named** 提供一个启动搜索其他站点的域信息的起点。没有了它们，**named** 只能知道它实际服务的域和它们的子域。

当 **named** 启动的时候，它就从根服务器之一重新载入线索。因此，只要您的线索文件里至少有一个有效、可达的根服务器，那么就没问题了。

线索文件经常叫做 **root.cache**。它包含了您向一台根服务器查询根域（“.”）中名字服务器的记录时所得到的响应。实际上，您可以就通过运行 **dig** 命令生成这个线索文件。

```
$ dig @f.root-servers.net . ns > root.cache
```

注意那有个点（.）。如果 f.root-servers.net 没有响应，那么您可以不指定一个特殊的服务器，再执行查询。

```
$ dig . ns > root.cache
```

输出的结果都近似，不过，您这次是从一台本地服务器的缓存，而不是一个权威来源，获得根服务器的列表。那个结果应该也没问题——即使您一两年都没有重新引导机器，或者重新启动您的名字服务器，它都会在根服务器记录的 TTL 到期时定期刷新根服务器的记录。

下面是缓存文件的样子（从 **dig @f.root-servers.net . ns** 的输出节录出来的）：

```
; <<>> DiG 9.3.0rc1 <<>> @f.root-servers.net . ns
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28797
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
```

www.TopSage.com

```
;; QUESTION SECTION:
;.                          IN      NS

;; ANSWER SECTION:
.                   518400  IN      NS      I.ROOT-SERVERS.NET.
.                   518400  IN      NS      J.ROOT-SERVERS.NET.
.                   518400  IN      NS      K.ROOT-SERVERS.NET.
.                   518400  IN      NS      L.ROOT-SERVERS.NET.
...
;; ADDITIONAL SECTION:
I.ROOT-SERVERS.NET. 3600000 IN      A       192.36.148.17
J.ROOT-SERVERS.NET. 3600000 IN      A       192.58.128.30
K.ROOT-SERVERS.NET. 3600000 IN      A       193.0.14.129
L.ROOT-SERVERS.NET. 3600000 IN      A       198.32.64.12
...
;; Query time: 3 msec
;; SERVER: 192.5.5.241#53(f.root-servers.net)
;; WHEN: Wed Jul  7 13:54:26 2004
;; MSG SIZE  rcvd: 436
```

注意，在第一组记录的开头都有个点（.），它可不是掉到纸上去的小污点，它定义了 NS 记录针对的域（即根域）。

BIND 9 把根服务器线索编译到了它的代码中，因此实际上不需要配置根区。不过，如果你提供了一个线索文件，BIND 9 还会使用它。我们建议您直接提供线索文件，政治已经介入了 DNS 领域，这使得根名字服务器和它们的 IP 地址更不稳定。

四、设置一个转发区

`forward` 类型的区将覆盖 **named** 对于一个特殊域的全转发设置（在 15.11.4 节介绍过）：

```
zone "domain_name" {
    type forward;
    forward only | first;
    forwarders { ip_addr; ip_addr; ... };
};
```

如果您的机构与其他机构或公司有战略合作关系，并且希望绕过标准的查询途径，直接把流量聚到该公司的名字服务器上，则可以使用 `forward` 区。

15.11.12 controls 语句

`controls` 语句规定了 **rndc** 如何控制一个正在运行的 **named** 进程。**rndc** 可以启动和停止 **named**，转储其状态，将其转入调试模式等。**rndc** 是一个网络程序，如果配置不正确，它就可能让 Internet 上的任何人把您的名字服务器搞糟。其语法是：

```
controls {
    inet ip_addr port ip-port allow { address_match_list } keys { key_list };
}
```

但如果没有出现 port 子句指定端口，那么 **rndc** 用来与 **named** 通信的默认端口是 953。

能远程控制您的名字服务器听起来既方便又危险。在 `allow` 子句中通过一个 `key` 项可以实现强验证功能，地址匹配列表中的密钥被忽略，必须在 `controls` 语句中的 keys 子句里显式地说明。

在 BIND 9 中，您可以使用 **rndc-confgen** 命令生成一个在 **rndc** 和 **named** 之间使用的验证密钥。设置它基本上有两种办法：您可以让 **named** 和 **rndc** 都参考同一个配置文件，以获得密

www.TopSage.com

钥（**/etc/rndc.key**），或者您也可以在每个程序各自的配置文件中告诉它密钥（**rndc** 的配置文件是**/etc/rndc.conf**，**named** 的配置文件是**/etc/named.conf**）。后一种选择更复杂，但是当 **named** 和 **rndc** 运行在不同的机器上时，就必须这么做。

没有出现 `controls` 语句的话，BIND 9 把地址匹配列表中默认为环回地址（loopbak），而且在**/etc/rndc.key** 里查找密钥。因为 BIND 9 必须要有强验证，所以，如果没有密钥的话，您不可以使用 **rndc** 命令来控制 **named**。这可能有点儿苛刻，但是想想看：即使只能从 localhost（127.0.0.1）调用 **rndc**，而且您的防火墙封锁了来自外界的这个地址，您仍然要信任所有的本地用户不会搞坏您的域名服务器。任何用户都能 **telnet** 到控制端口并键入“stop”——相当有效的一种拒绝服务攻击。

下面是在采用 256 位密钥时 **rndc-confgen** 输出（到标准输出设备）的一个例子。我们选择 256 位是因为它在本书的一行里放得下。在正常情况下，您会把输出重定向到**/etc/rndc.conf**。输出最后的注释给出了需要加到 **named.conf** 中的行，它们能让 **named** 和 **rndc** 一起工作。

```
$ ./rndc-confgen -b 256
# Start of rndc.conf
key "rndc-key" {
    algorithm hmac-md5;
    secret "orZuz5amkUnEp52zlHxD6cd5hAClOGsG/elP/dv2IY=";
};

options {
    default-key "rndc-key";
    default-server 127.0.0.1;
    default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
#       algorithm hmac-md5;
#       secret "orZuz5amkUnEp52zlHxD6cd5hAClOGsG/elP/dv2IY=";
# };
#
# controls {
#       inet 127.0.0.1 port 953
#               allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

用于产生一个共享配置文件的“自动模式”要更简单。以 root 身份运行，**rndc-confgen -a -b 256** 就会生成文件**/etc/rndc.key**，其中包括：

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "laGbZj2Cobyc0m/jFVNCu8OJzsLKNH+CCb2JCWY6yJw=";
};
```

这个文件的模式为 600，它的属主应该是 **named** 用户或者 root。接下来，可以用 `include` 语句把它加到 **named.conf** 文件中去。

15.11.13 分离式 DNS 和 view 语句

许多站点想要让网络从内部看起来不同于从外部 Internet 上看到的样子。例如，您可能向内部用户公开一个区的所有主机，但是却限制外界，使其只能看到几台众所周知的服务器。或者，您可能想

www.TopSage.com

让内外双方看到的都相同，但却给内部用户提供更多的（或者不同的）记录。例如，从外界来看，用于路由邮件的 MX 记录可能指向了一台枢纽机器，而从内部用户的角度来看，MX 记录却指向了各台工作站。

对于在其内部网络上使用了 RFC1918 私用 IP 地址空间的站点来说，配置分离式 DNS（split DNS）格外有用。例如，对 IP 地址 10.0.0.1 所关联的主机名的查询决不会从全球的 DNS 系统中得到回答，但是在本地网络的环境下，却会具有意义。在根名字服务器收到的查询之中，有 4%~5%要么是来自私用地址范围内的 IP 地址，要么是要查询这些地址中的某个地址的域名。这两种情况都不会得到回答，出现这两种情况都是因为配置错误，要么是 BIND 的分离式 DNS 配错了，要么是 Microsoft 的“域（domain）”配错了。参考 12.4.6 节了解有关私有地址空间的知识。

view 语句将一个用于控制哪些客户机能看到哪个 view（视图）的访问列表、用于 view 中所有区的一些选项、最后还有区本身打包在一起。语法是：

```
view view-name {
    match-clients { address_match_list } ;
    view_option; ...
    zone_statement; ...
} ;
```

match-clients 子句控制谁能看到 view（视图）。view 是按顺序来处理的，所以要先加上限制最严格的 view。在不同 view 中的区可以同名。view 是一个“全有全无”命题，如果您使用了它们，那么 **named.conf** 文件中所有的 zone 语句都必须在 view 中出现。

下面是一个从 BIND 9 的文档中摘出来的例子。在同一个区定义了两个 view，但是用的数据不一样。

```
view "internal" {
    match-clients { our_nets; };        // only internal networks
    recursion yes;                      // internal clients only
    zone "example.com" {                // complete view of zone
        type master;
        file "example-internal.db";
    };
};

view "external" {
    match-clients { any; };             // allow all queries
    recursion no;                       // but no recursion
    zone "example.com" {                // only "public" hosts
        type master;
        file "example-external.db";
    }
};
```

如果 view 的次序反了，那么谁都看不到内部的 view 了。内部主机在它们碰到内部的 view 之前，就先会匹配外部的 view 的 match-clients 子句里的 any 值。我们的第二个 DNS 配置示例（15.12.2 节）提供了更多的 view 的例子。

15.12 BIND 配置举例

既然我们已经详细解释了 **named.conf** 的内容，就让我们看几个完整的例子吧。在接下来的几小节里，我们将给出 3 个样板配置：

- localhost 区；

www.TopSage.com

- 一家小型的安全公司，使用分离式 DNS（split DNS）；
- 专业的系统：isc.org，（Internet Systems Consortium，Internet 系统联盟）。

15.12.1 localhost 区

地址 127.0.0.1 指代主机自身，它应该被映射到域名“localhost.”[16]。某些站点把这个地址仅映射为“localhost.*localdomain*”，有些两者都映射。对应的 IPv6 地址是::1。

如果您忘记配置 localhost 区，那么您的站点最终会向根服务器查询本地主机的信息。根服务器目前接收到如此众多这样的查询，以至于操作人员正考虑在根级添加 localhost 和 127.0.0.1 之间的一个普通映射。根据 2004 年 3 月对位于旧金山名为 F 的根服务器的测量结果，localhost 是被查询最多的第 3 个域名，仅次于 arpa 和 com。给一台繁忙的根服务器造成了那么多无用的查询。

localhost.*localdomain* 这个名字的正向映射可以在该域的正向区文件中定义，或者可以在它自己的文件里定义。每台服务器，甚至是一台缓存服务器，通常都是它自己的反向 localhost 域的主服务器。

下面是从配置了 localhost 的 **named.conf** 文件看到的配置行：

```
zone "localhost" {// localhost forward zone
    type master;
    file "for/localhost";
    allow-update { none; };
};

zone "0.0.127.in-addr.arpa" {// localhost reverse zone
    type master;
    file "rev/127.0.0";
    allow-update { none; };
};
```

对应的正向区文件是 **for/localhost**，其内容为：

```
$TTL 30d
; localhost.
@       IN  SOA   localhost. postmaster.localhost. (
                      1998050801     ;serial
                      3600           ;refresh
                      1800           ;retry
                      604800         ;expiration
                      3600 )         ;minimum

            NS    localhost.
            A     127.0.0.1
```

反向区文件 **rev/127.0.0** 为：

```
$TTL 30d
; 0.0.127.in-addr.arpa
@       IN  SOA   localhost. postmaster.localhost. (
                      1998050801     ;serial
                      3600           ;refresh
                      1800           ;retry
                      604800         ;expiration
                      3600 )         ;minimum

            NS    localhost.
1           PTR   localhost.
```

[16] 实际上，整个 A 类网络 127/8 都指向 localhost，但是大多数人都只用 127.0.0.1。

www.TopSage.com

localhost 地址（127.0.0.1）的反向映射永远不会更改，因此超时时间可以很长。注意用日期编码的序列号，该文件最后一次更改是在 1998 年。还要注意 localhost 域只会列出主名字服务器。这里的@的含义是“0.0.127.in-addr.arpa.”。

15.12.2 一家小型的安全公司

我们举的第一个真实的例子用于一家小公司，这个公司主做安全咨询。他们在最新版的 RHEL 上运行 BIND 9，使用视图（view）来实现一个分离式 DNS（split DNS）系统，在这个系统上，内外用户看到的主机数据是不一样的。他们还在内部使用了私用的地址空间；有关那些地址的查询一定不能漏到 Internet 上去，搞乱全球的 DNS 系统。下面是他们的 **named.conf** 文件，我们对它的格式重新做了调整，并且加入了一点儿注释：

```
options {
    directory "/var/domain";
    version "root@atrust.com";
    allow-transfer {82.165.230.84; 71.33.249.193; 127.0.0.1; };
    listen-on { 192.168.2.10; 192.168.2.1; 127.0.0.1; 192.168.2.12; };
};

include "atrust.key";                          // defn of atkey in mode 600 file

controls {
    inet 127.0.0.1 allow { 127.0.0.1; } keys { atkey; };
};

view "internal" {                              // internal view

    match-clients { 192.168.0.0/16; 206.168.198.192/28; 172.29.0.0/24; };
    recursion yes;

    zone "." IN {                              // root hints zone
        type hint;
        file "named.cache";
    };
    zone "localhost" IN {                      // localhost forward zone
        type master;
        file "localhost.forw";
        allow-update { none; };
    };
    zone "0.0.127.in-addr.arpa" IN {           // localhost reverse zone
        type master;
        file "localhost.rev";
        allow-update { none; };
    };
    zone "atrust.com" IN {                     // internal forward zone
        type master;
        file "internal/atrust.com";
    };
    zone "1.168.192.in-addr.arpa" IN {         // internal reverse zone
        type master;
        file "internal/192.168.1.rev";
        allow-update { none; };
    };                                         // lots more reverse zones omitted
    zone "atrust.us" {                         // slave in internal view
        type slave;
        file "internal-slave/atrust.us";
```

www.TopSage.com

```
        masters { 127.0.0.1; };
    };
    zone "atrust.org" {
        type slave;
        file "internal-slave/atrust.org";
        masters { 127.0.0.1; };
    };                                  // lots more zones omitted
};      // end of internal view

view "world" {                          // external view

    match-clients { any; };
    recursion no;

    // zone statements for dot and localhost as above, omitted for brevity

    zone "atrust.com" {                 // external forward zone
        type master;
        file "world/atrust.com";
        allow-update { none; };
    };
    zone "189.173.63.in-addr.arpa" {    // external reverse zone
        type master;
        file "world/63.173.189.rev";
        allow-update { none; };
    };
    zone "atrust.us" {                  // master in world view
        type master;
        file "world/appliedtrust.com-aggregate";
        allow-update { none; };
    };
    zone "atrust.org" {                 // note that the file is the same
        type master;
        file "world/appliedtrust.com-aggregate";
        allow-update { none; };
    };                                  // lots more atrusty zones omitted
    zone "admin.com" {                  // master zones only in world view
        type master;
        file "world/admin.com";
        allow-update { none; };
    };                                  // lots more master zones omitted
    zone "eldoraskipatrol.org" {        // slave zones only in world view
        type slave;
        file "world-slave/eldoraskipatrol.org";
        masters { 192.231.139.1; };
    };                                  // lots more slave zones omitted
}; // end of external view
```

atrust.key 文件包含了名为“atkey”的密钥的定义。

```
key "atkey" {
    algorithm hmac-md5;
    secret "shared secret key goes here";
};
```

这些区按照它们的 view（视图）和类型（master/slave）来组织，区数据文件的命名习惯反映了同样的组织方式。这个服务器从内部看（“internal” view）是递归的，包括了所有的本地主机，其中许多使用了私有地址。从外部看（“world” view）是非递归的，只包括几台从 atrust.com 选出的主机，它们提供的外部区不是主（master）DNS 设备就是从（slave）DNS 设备。

www.TopSage.com

大家网 TopSage.com

下面是文件 **internal/atrust.com** 和 **world/atrust.com** 的片段。首先是内部的文件：

```
; atrust.com - internal file
$TTL 86400
$ORIGIN com.
atrust          3600   SOA   ns1.atrust.com. trent.atrust.com. (
                                  2004012900 10800 1200 3600000 3600 )
                3600   NS    NS1.atrust.com.
                3600   NS    NS2.atrust.com.
                3600   MX    10 mailserver.atrust.com.
                3600   A     66.77.122.161

$ORIGIN atrust.com.
ns1                    A     192.168.2.11
ns2                    A     66.77.122.161
www                    A     66.77.122.161
mailserver             A     192.168.2.11
exchange               A     192.168.2.100
secure                 A     66.77.122.161
bark                   A     192.168.2.10
superg                 A     192.168.1.249
at-dmz-gw              A     192.168.1.254
at-external-gw         A     192.168.2.254
at-external-outside-gw A     206.168.198.220
indras-gw              A     206.168.198.222
; dhcp host range
dhcp-0-hardwire  IN    A     192.168.1.64
dhcp-1-hardwire  IN    A     192.168.1.65
; ...

; booklab "subdomain", note that this is a subdomain but not a subzone

redhat.booklab      IN    A     192.168.10.1
redhat-ent.booklab  IN    A     192.168.10.2
debian.booklab      IN    A     192.168.10.3
fedora.booklab      IN    A     192.168.10.4
freebsd.booklab     IN    A     192.168.10.5
suse.booklab        IN    A     192.168.10.6
macos.booklab       IN    A     192.168.10.7
solaris.booklab     IN    A     192.168.10.8
```

下面是文件 **world/atrust.com** 里对同一个域 **atrust.com** 的外部 view：

```
; atrust.com - external file
$TTL 57600
$ORIGIN .
atrust.com                SOA   ns1.atrust.com. trent.atrust.com. (
                                   2004020400 10800 1200 3600000 3600 )
                          NS    NS1.atrust.com.
                          NS    NS2.atrust.com.
                          MX    10 mailserver.atrust.com.
                          A     66.77.122.161
ns1.atrust.com.           A     206.168.198.209
ns2.atrust.com.           A     66.77.122.161

$ORIGIN atrust.com.
www                       A     66.77.122.161
mailserver                A     206.168.198.209
bark                      A     206.168.198.209
secure                    A     66.77.122.161
```

www.TopSage.com

```
; reverse maps
exterior1                   A     206.168.198.209
209.198.168.206             PTR   exterior1.atrust.com.
exterior2                   A     206.168.198.213
213.198.168.206             PTR   exterior2.atrust.com.
exterior3                   A     206.168.198.220
220.198.168.206             PTR   exterior3.atrust.com.
exterior4                   A     206.168.198.210
210.198.168.206             PTR   exterior4.atrust.com.
```

注意，同一台机器需要多个名字的时候，可以用更多的 A 记录而不是 CNAME 记录来表示它们。在客户机解析名字 www.atrust.com 的时候，如果用 CNAME 记录把 www 映射到了实际的服务器上，那么会多做一次查询，因此有可能使得查询 IP 地址所花的时间加倍。

在这些区文件中的 TTL 被默认设置为 16 小时（57 600 秒）。对于内部的区，TTL 是 1 天（86 400 秒）。区文件内的大多数记录都没有指派一个明确的 TTL 值。TTL 是可有可无的，区文件任何一行的记录类型前面都可以加上它。

在外部区文件末尾有一些奇怪的 PTR 记录，它能让 atrust.com 的 ISP 授权得到一小段地址空间的反向映射。在 ISP 站点上的 CNAME 记录能让这种 CNAME 的扩展起作用，参考 15.7.8 节了解更多的信息。

15.12.3 ISC

ISC 既是 BIND 的开发者，也是 BIND 的维护者，它还负责根名字服务器 F 的运行。ISC 也是一个 TLD 服务商，提供大约 60 个顶级域名的服务。这也是我们称之为专家的原因！

下面是从 ISC 的配置文件中节选出来的片段。注意，他们正同时使用着 IPv4 和 IPv6。ISC 还使用 TSIG 加密来做区传送时主从服务器之间的身份验证。`transfer-source` 选项确保了从服务器发出的区传送请求的源 IP 地址符合主服务器上 `allow-transfer` 语句的规定。

named.conf 文件：

```
// isc.org TLD name server

options {
     directory "/var/named";
     datasize 1000M;
     listen-on { 204.152.184.64; };
     listen-on-v6 { 2001:4f8:0:2::13; };
     recursion no;
     transfer-source 204.152.184.64;
     transfer-source-v6 2001:4f8:0:2::13;
};

// rndc key
key rndc_key {
     algorithm hmac-md5;
     secret "<secret>";
};

// TSIG key for name server ns-ext
key ns-ext {
     algorithm hmac-md5;
     secret "<secret>";
};

server 204.152.188.234 { keys { ns-ext; }; };
```

www.TopSage.com

```
controls {
    inet 204.152.184.64 allow { any; } keys { rndc_key; };
};

include "inf/named.zones";          // root, localhost, 127.0.0.1, ::1
include "master.zones";             // zones we master
include "slave.zones";              // lots of slaves
```

这几个 include 语句让 **named.conf** 文件保持简短和整洁。如果您要提供多个区的名字服务，那么可以考虑将配置分成与此相似的适当大小。更重要的是，要建立其文件系统的层次结构，以免在一个目录下有上千个区文件。现代的 Linux 发行版本处理大目录的效率很高，但是它们管理起来可很费劲儿。

下面是文件 **master.zones** 中的内容：

```
zone "isc.org" {                    // ISC
    type master;
    file "master/isc.org";
    allow-update { none; };
    allow-transfer { none; };
};
zone "sfo2.isc.org" {               // ISC
    type master;
    file "master/sfo2.isc.org";
    allow-update { none; };
    allow-transfer { none; };
};

// lots of zones truncated
```

而 **slaves.zones** 文件的内容为：

```
zone "vix.com" {
    type slave;
    file "secondary/vix.com";
    masters { 204.152.188.234; };
};

zone "cix.net" {
    type slave;
    file "secondary/cix.net";
    masters { 204.152.188.234; };
};
```

通过上面的学习，掌握了 DNS 的背景和 BIND 的配置知识之后，现在我们就更侧重于讨论在您的站点上跑一个名字服务器时的运行问题，包括区文件的维护、安全事务、测试和调试。

15.13 启动 named

Linux 发行版本上带的 BIND 软件情况，取决于具体的发行版本，以及在安装发行版本时所做的选择。如果 BIND 似乎没有装，那么必须获得并安装 BIND 软件包。这个软件包也会把通过 **init** 运行的 **named** 的启动脚本安装好：对于 RHEL、Fedora 和 SUSE 是**/etc/init.d/named**，对于 Debian 和 Ubuntu 是**/etc/init.d/bind9**。

named 在系统引导时启动并持续运行。要控制一个正在运行的 **named**，您可以使用命令行接口 **rndc**，我们在 15.16.4 节将更详细地介绍这个工具。

www.TopSage.com

有时候，启动 **named** 的 **init** 脚本提供了额外的启动方式（例如，**reload**），专门供系统管理员使用。不过，在有些 Linux 发行版本上的实现存在问题，还是使用 rndc 比较简单。我们建议您把启动脚本留给操作系统和 **init** 使用，而在 **named** 已经启动之后用 **rndc** 对它进行控制。

named 使用 syslog，所以在 **named** 之前应该先启动 **syslogd**。不要使用 **inetd** 或者 **xinetd** 来管理 **named**，它会在每次需要用 **named** 的时候重新启动它，因而降低了反应速度，而且会阻止形成有用的缓存。有些站点使用一直运行的脚本，在其中把 **named** 的启动命令放在一个死循环里，以此防止 **named** 进程死掉，参考 15.9.3 节中给出的例子。BIND 软件发布包括一个叫做 **nanny** 的脚本，它能达到类似的目的。参考 29.3.2 节了解有关 **inetd** 的更多知识。

15.14 更新区文件

当您对某个区做了一次更改时（例如，添加或删除一台主机），主服务器上的数据文件必须更新。您还必须增加区的 SOA 记录中的序列号，然后运行 **rndc reload**，发信号通知 **named** 读取这些更改。您还可以杀死并重新启动 **named**，但是这一操作会造成其他域的缓存数据被丢弃。

BIND 的早期版本使用信号和 **kill** 命令来控制 **named**，但是正当开发人员开始用完了信号的编号时，就出现了 **rndc** 并解决了所有问题。历史上在 BIND 中使用的信号（除了用来重新读取配置文件、重新载入区和重新启动 **named** 的 HUP 信号，以及杀死 **named** 的 TERM 信号和 INT 信号）都已经从 BIND 9 中删除，所以我们建议坚持使用 **rndc**。

因为 `notify` 选项是默认打开的，所以更新后的区数据会马上传送到从服务器。如果您已经无意中关闭了这个选项，那么您的从服务器直到 *refresh*（刷新）时间以后才能得到这些更新，这个时间在区的 SOA 记录中设置（一般是 1~6 个小时）。如果您希望在关闭了 `notify` 选项后更及时地进行更新，那么从服务器上的 **rndc reload** 命令能让它检查主服务器，查看数据是否已经更改，并请求区传送。

当您改了一个主机名或 IP 地址时，不要忘记修改正反向区。忘记修改反向文件会留下一些神秘的错误：有些命令工作正常，有些命令工作不正常。

改了数据文件但是忘记改序列号只会使改动在主服务器上生效（经过一次重新加载），但是不会在从服务器上生效。

编辑属于从服务器的数据文件是不合适的。这些文件应由 **named** 维护，系统管理员不应该干预它们。只要您不做更改，查看这些文件还是可以的。

BIND 允许通过在 RFC2136 中指定的编程 API 对区进行更改。这项特性称为动态更新，它对于类似 DHCP 这样的自动配置协议来说是必要的。动态更新机制将在 15.14.2 节介绍。

15.14.1 区传送

DNS 服务器通过一种称为区传送的机制保持同步。最初的 DNS 规范（和 BIND 4）要求一次传送所有的区数据。增量更新最终在 RFC1995 中定义，并在 BIND 8.2 中得以实现。最初的区传送和增量风格的区传送有时分别叫做 AXFR 和 IXFR。一旦配置完毕，它们就应该是等价的。

希望刷新自身数据的从服务器请求从主服务器作一次区传送，并在磁盘上保存区数据的备份副本。如果通过比较序列号（不是实际的数据）判断得知主服务器上的数据没有更改，则不发生任何更新，备份文件只是稍作修改（即把它们的修改时间设置为当前时间）。

区传送在端口 53 上使用 TCP 协议，并且通过 syslog 用“transfer-*”类记录日志信息。根据 IETF 的规定，IXFR 既可以使用 TCP 也可以使用 UDP，但是 BIND 只在 TCP 上实现了它。

发送和接收服务器在区传送过程中仍然可以回答名字查询。只有在传送完毕之后，从服务器才开

www.TopSage.com

始使用新数据。BIND 8 实际上调用了一个独立的 **named-xfer** 程序来执行传送，但是 BIND 9 的 **named** 则直接处理传送。

当区变得十分庞大（如 com）或是采用了动态更新（请参考下一节），变更部分相对于整个区的大小来说通常比较小。通过 IXFR，可以只发送变更部分（除非它们比整个区都大，此时执行常规的 AXFR 传送）。IXFR 机制类似于 **patch** 程序，因为它把新老差异应用到老数据库上，使老数据库与新数据库同步。

在 BIND 9 中，IXFR 是配置成动态更新的任何区域的默认值，**named** 总是会维护一个叫做 *zonename*.**jnl** 的事务日志。可以针对个别对等服务器在 server 语句中设置选项 `provide-ixfr` 和 request-iexfr。`provide-ixfr` 为该服务器在其中是主服务器的区域启用或禁用 IXFR 服务。`request-ixfr` 选项为该服务器在其中是从服务器的区域要求 IXFR。

```
provide-ixfr yes ;              # in BIND 9 server statement
request-ixfr yes ;              # in BIND 9 server statement
```

从 BIND 9.3 开始，named 能够让从服务器使用 IXFR 传送手工编辑的数据。使用 zone 语句选项 `xifr-from-differences` 就可以启用这个功能，参考 15.11.4 节的内容。

IXFR 要求区文件按一种规定的顺序进行排序。named 代替您完成这项任务，但是这一要求会让 IXFR 做出一种权衡。在服务器上使用更多的内存和 CPU，才能换来网络流量的减少。

人们做了许多工作，以确保服务器在一次 IXFR 期间如果崩溃也不会在区中留下垃圾数据。对不支持 IXFR 的服务器的请求，IXFR 会自动转为标准的 AXFR 区传送。

15.14.2 动态更新

DNS 系统是建立在名字到地址的映射相对稳定而且不经常更改这一前提之上的。不过，在机器引导并加入网络时用 DHCP 来动态分配 IP 地址的网点则肯定破坏了这一规则。传统的解决方案有两种：向 DNS 数据库添加若干个一般项，或者持续不断地编辑 DNS 文件。对于许多网点来说，这两种解决方案都不能令人满意。

任何使用拨号 ISP 的人都应该对第一种解决方案很熟悉。此时的 DNS 配置看起来有点类似这样：

```
dhcp-host1.domain.      IN  A   192.168.0.1
dhcp-host2.domain.      IN  A   192.168.0.2
...
```

尽管这是一个简单的解决方案，但是它意味着主机名永久性地与特定的 IP 地址相关，因此每当计算机接收到一个新的 IP 地址时，就要更改主机名。在这种环境下，基于主机名的日志或安全措施会变得十分困难。

BIND 最新版本中的动态更新特性提供了另一种解决方案。它让 DHCP 守护进程通知 BIND 它所作的地址分配情况，因而可以随时更新 DNS 数据库的内容。动态更新可以添加、删除或修改资源记录。调节动态更新的粒度是区。在动态更新的同时，一个日志文件（*zonename*.**jnl**）被保留下来，以防服务器崩溃。

您必须先停止正在进行的动态更新，才能手工编辑一个动态更新的区。**rndc freeze** *zone* 或者 **rndc freeze** *zone class view* 就可以做到这一点。这两条命令将上述日志文件和磁盘上的主区文件做同步，然后删除这个日志文件。接下来，您就可以手工编辑区文件。当然，区文件的最初格式将被 **named** 破坏——这个文件看起来像由 **named** 为从服务器所留的那些区文件。

在区被冻结的时候，对该区做动态更新将被拒绝，使用 **rndc thaw** 命令，带上和冻结区时使用的

www.TopSage.com

参数一样的参数，就可以从磁盘重新载入区文件，重新启用动态更新。

BIND 9 带的程序 **nsupdate** 提供了一个用于动态更新的命令行界面。它以批处理方式运行，从键盘或者文件接受命令。用一个空行或者 send 来结束一次更新，并把变化发送给服务器。两个空行表示输入结束。命令语言中包括一个原始的“if”语句，可以构造这样的表达式“如果 DNS 中不存在这个主机名，那么就添加它”。**nsupdate** 肯定有这样的功能，您可以要求一个名字存在或是不存在，或者要求一组资源记录存在或是不存在。

例如，下面是一个简单的 **nsupdate** 脚本，它加入一台新主机，如果现有的一台主机没有用昵称，它也会给它取个昵称。尖括号的提示符是由 **nsupdate** 生成的，不是命令脚本的内容。

```
$ nsupdate
> update add newhost.cs.colorado.edu 86400 A 128.138.243.16
>
> prereq nxdomain gypsy.cs.colorado.edu
> update add gypsy.cs.colorado.edu CNAME evi-laptop.cs.colorado.edu
```

DNS 的动态更新有点儿让人不放心。它们有可能会提供一些不受控的写操作，可以访问到您的重要系统数据。不要试着用 IP 地址来做访问控制——它们太容易假冒了。采用有“共享秘密”密钥的 TSIG 验证比较好，它能用而且容易配置。这两种方式 BIND 9 都支持：

```
$ nsupdate -k keydir:keyfile
```

或者

```
$ nsupdate -y keyname:secretkey
```

因为口令是在命令行上提供的，在此刻运行 **w** 或者 **ps** 命令的任何人都能看到它。出于这个原因，要用**-k** 形式。要了解有关 TSIG 的详情，可参考 15.15.3 节。

在 **named.conf** 文件中用 `allow-update` 或者 `update-policy` 子句可以启动一个区的动态更新。`allow-update` 授权按照源地址或者基于密钥的验证来更新任何记录。`update-policy` 是 BIND 9 实现的一个扩展，可以按照主机名或者记录类型对更新做更细粒度的控制。它要基于密钥的验证。两者都是 zone 语句的选项。

用 `update-policy` 来让客户机更新它们的 A 记录或者 PTR 记录，但不改变 SOA 记录、NS 记录或者 KEY 记录。您甚至可以用它来让主机只更新它自己的记录。它的参数可以让您显式地给出名字，用作子域名、通配符或者关键字 `self`，self 为机器访问它们自己的记录设置了一般性的策略。资源记录是由 class（类）和 type（型）来标识的。`update-policy` 规则的语法是：

```
update-policy ( grant | deny ) identity nametype name [types] ;
```

identity 是授权更新所需要的密钥名。*nametype* 可以取 4 个值之一：`name`、`subdomain`、`wildcard` 或者 `self`。name 是要被更新的区、types 是可以被更新的资源记录类型。如果没有指定类型，那么除了 SOA、NS、RRSIG 和 NSEC 之外的所有资源类型都会被更新。

下面是一个例子：

```
update-policy { grant dhcp-key subdomain dhcp.cs.colorado.edu A } ;
```

这一配置允许知道密钥 `dhcp-key` 的任何人更新 dhcp.cs.colorado.edu 子域中的地址记录。这条语句应该出现在主服务器的 **named.conf** 文件中，位于 dhcp.cs.colorado.edu 的 `zone` 语句之下。还要有一条 `key` 语句来定义 `dhcp-key`。

下面的片段取自于科罗拉多大学计算机科学系的 **named.conf** 文件，它使用 `update-policy` 语句让上系统管理课的同学们更新他们自己的子域，但又不会弄乱 DNS 环境中的其他部分。

www.TopSage.com

```
// saclass.net
zone "saclass.net" in {
    type master;
    file "saclass/saclass.net";
    update-policy {
        grant feanor_mroe. subdomain saclass.net.;
        grant mojo_mroe. subdomain saclass.net.;
        grant dawdle_mroe. subdomain saclass.net.;
        grant pirate_mroe. subdomain saclass.net.;
        // and lots more
    };
...
```

15.15 安全问题

过去，Internet 规模小、很友好，而且使用者大多是计算机迷。现如今，Internet 既是一个充满敌意的环境，又是一种重要的基础设施。这一节介绍几种与安全相关的话题，介绍的方式可能显得有些偏执。遗憾的是，在今天的 Internet 上，这些话题和警告非常必要。

DNS 开始本打算作为一种开放系统，但是它已经逐渐变得越来越安全了——或者至少是越来越能变得安全了。在默认情况下，Internet 上任何人可以用类似 **dig**、**host** 或 **nslookup** 这样的工具程序一步步进行查询来调查您的域。在某些情况下，他们能转储您的整个 DNS 数据库。

为了解决这样的安全漏洞，BIND 现在支持基于主机和网络地址或者基于加密验证的多种类型的访问控制。表 15.11 归纳了 **named.conf** 中配置的安全特性。在章节一列指出了可以在本书的哪些地方可以找到更多信息。

表 15.11 named.conf 中的安全特性

特 性	语 句	章 节	指定的内容
acl	各种语句	15.11.5	访问控制列表
allow-query	options, zone	15.11.4	谁能查询区或者服务器
allow-recursion	options	15.11.4	谁能执行递归查询
allow-transfer	options, zone	15.11.4	谁能请求区传送
allow-update	zone	15.11.11	谁能执行动态更新
blackhole	options	15.11.4	完全忽略哪些服务器
bogus	server	15.11.8	从不应该查询哪些服务器
update-policy	zone	15.12.2	允许做什么样的更新

named 可以用非特权的 UID 在 **chroot** 的环境中运行，因此把风险降到了最低。它可以使用事务签名来控制动态更新或者区传送，当然，它还支持整个 DNSSEC。这些主题将在接下来的几小节里进行讨论。

15.15.1 再谈访问控制列表

ACL 是有名字的地址匹配列表，它可以作为参数出现在诸如 `allow-query`、`allow-transfer` 和 `blackhole` 这样的语句中。它们的基本语法在 15.11.5 节介绍过了。ACL 能以各种不同方式加强 DNS 的安全。

每个站点至少应该有一个假地址的 ACL 和一个本地地址的 ACL。

www.TopSage.com

例如：

```
acl bogusnets {                 // ACL for bogus networks
    0.0.0.0/8 ;                 // default, wild card addresses
    1.0.0.0/8 ;                 // reserved addresses
    2.0.0.0/8 ;                 // reserved addresses
    169.254.0.0/16 ;            // link-local delegated addresses
    192.0.2.0/24 ;              // sample addresses, like example.com
    224.0.0.0/3 ;               // multicast address space
    10.0.0.0/8 ;                // private address space (RFC1918)[17]
    172.16.0.0/12 ;             // private address space (RFC1918)
    192.168.0.0/16 ;            // private address space (RFC1918)
} ;

acl cunets {                    // ACL for University of Colorado networks
    128.138.0.0/16 ;            // main campus network
    198.11.16/24 ;
    204.228.69/24 ;
};
```

在配置文件的全局 options 部分中，可以包括：

```
allow-recursion { cunets; } ;
blackhole { bogusnets; } ;
```

把区传送限制在合法的从服务器范围内执行也是个不错的主意。采用一个 ACL 将使事情变得既美观又整洁。

```
acl ourslaves {
    128.138.242.1 ;             // anchor
    ...
} ;
acl measurements {
    198.32.4.0/24 ;             // bill manning's measurements, v4 address
    2001:478:6:0::/48 ;         // bill manning's measurements, v6 address
} ;
```

用类似下面的一行实现实际的限制：

```
allow-transfer { ourslaves; measurements; } ;
```

传送只能限制到我们自己的从服务器以及两台 Internet 测量项目的机器，这个测量项目遍历反向 DNS 树判断 Internet 的大小以及配置错误的服务器的比例。用这种方法限制传送，可以使其他站点不可能用 **nslookup**、**dig** 或 **host** 转储您的整个数据库（参见 15.16.7 节）。

例如，caida.org 使用了一个 ACL 限制只有其二级服务器才能做区传送，从 jungle.caida.org 使用 **dig** *@server domain-to-dump* **axfr** 可以成功做区传送，但是从 gypsy.cs.colorado.edu 则不行：

```
jungle$ dig @rommie.caida.org caida.org axfr
; <<>> DiG 8.3 <<>> @rommie.caida.org caida.org axfr
; (1 server found)
$ORIGIN caida.org.
@                       4H  IN  SOA     @ postmaster (
                                        200406300       ; serial
                                        1H              ; refresh
                                        30M             ; retry
                                        1W              ; expiry
```

[17] 如果您用到了私用地址，而现在是在配置内部的 DNS 服务器，那就不要让它们成了 bogus。

www.TopSage.com

)

...

gypsy$ **dig @rommie.caida.org caida.org axfr**

; <<>> DiG 9.2.4 <<>> @rommie.caida.org caida.org axfr
;; global options: printcmd
; Transfer failed.

当然，您应该仍然通过路由器的访问控制列表以及每台主机上常规的安全保护机制，在更低层上保护网络。如果不可能使用这些手段，那么除了您能紧密监控的一台网关机器之外，在其他地方都拒收 DNS 包。

15.15.2 限制 named

为了限制侵入您的服务器的人所能造成的破坏，可以在已经 **chroot** 过的环境中运行 **named** 并且/或者把它作为一个非特权用户程序来运行。**-t** 标志指定要 **chroot** 到的目录，**-u** 标志指定运行它的 UID。例如，命令

```
# named -u 53 -t /var/named
```

会以 UID 53 和根目录**/var/named** 来启动 **named**。

如果黑客们侵入了您的 **named**，则无论 **named** 以何种用户身份运行，他们都有可能获得系统的访问权限。如果这个用户为 root，而您没有使用 **chroot** 过的环境，那么这样的漏洞破坏性可能很大。许多站点怕麻烦，不使用**-u** 和**-t** 标志，但是当一项新的安全漏洞公布出来的时候，他们必须赶在黑客之前升级系统。

chroot 目录不能是一个空目录，因为它必须包含 **named** 运行通常所需的所有文件：**/dev/null**、区文件、**named.conf**、syslog 目标文件和 syslog UNIX 域套接口、**/var** 等。BIND 9 的 **named** 在加载了所有的库之后，会执行系统调用 **chroot**，所以再没有必要把共享库复制到 **chroot** 目录下了。



Fedora 默认安装一个用 **chroot** 改变过根目录的 **named**。

15.15.3 使用 TSIG 和 TKEY 保障服务器与服务器之间通信的安全

尽管 IETF 正在制订 DNSSEC（在下一节中介绍）的规范，但 IETF 还开发了一种更简单的机制，称为 TSIG（RFC2845），通过使用“事务签名（transaction signature）”来保障服务器之间的通信安全。基于事务签名的访问控制比基于 IP 源地址的访问控制更安全。TSIG 能保护主服务器和它的从服务器之间区传送的安全，并且能实现安全的动态更新。

TSIG 签名只用在服务器之间的消息和响应上，而不用在服务器和解析器之间。事务签名要验证双方的身份，证实数据没有被窜改过。签名在接收到一个包时被检查，然后被丢弃，它们不会被缓存也不会成为 DNS 数据的一部分。

TSIS 使用一种对称加密方案。也就是说，加密密钥与解密密钥相同。这个密钥称为“共享秘密”密钥。尽管 TSIG 规范可以采用多种加密方法，但是 BIND 仅实现了一种，即 HMAC-MD5 算法。

您应该为希望进行安全通信的每对服务器使用不同的密钥。TSIG 比公钥加密方法的计算开销少得多，但是仅适合于有少量成对通信服务器的本地网络。它不能扩展到全球的 Internet。

BIND 的 **dnssec-keygen**[18]工具程序为每对服务器产生一个密钥。例如，要为两个服务器 *serv1* 和

[18] 这条命令在 BIND 8 中叫做 **dnskeygen**。

www.TopSage.com

serv2 产生“共享秘密”的主机密钥，则使用

```
# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST master-slave1
```

命令创建一个 128 位密钥并生成两个文件：**K*serv1-serv2*.+157+09068.private** 和 **K*serv1-serv2*.+157+09068.key**。**157** 代表 HMAC-MD5 算法，**09068** 是一个随机数[19]，用作密钥标识符，以防您为同一对服务器用了多个密钥。**.private** 文件看上去像是这样：

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: jxopbeb+aPc71Mm2vc9R9g==
```

而**.key** 文件像这样：

```
master-slave1. IN KEY 512 3 157 jxopbeb+aPc71Mm2vc9R9g==
```

这两个文件的模式都应该是 600，而且属主应该是 **named** 用户。

注意，在文件名上和**.key** 文件的内容里，都已经在 master-slave1 参数字符串后面加了一个点(.)。这个习惯的原因是，在别的地方，密钥名必须是完整的域名，因此必须以一个点（.）结尾。

实际上，您根本不需要**.key** 文件——它之所以会产生，是因为 **dnssec-keygen** 程序也生成公钥对，其中的公钥（**.key** 文件）是作为 KEY 资源记录被插入到 DNS 区文件里的。KEY 记录里的 512 不是密钥的长度，而是一个标志位，指出该记录是一个 DNS 密钥记录。

在经过如此复杂的一个过程之后，生成的密钥实际上只是一个长随机数，您可能会对此感到失望。您可以写下一个长度正确（能被 4 整除）的 ASCII 字符串，假装它是一个 base-64 密码，或者使用 **mmencode** 对一个随机串进行加密，手工生成密钥。您创建密钥的方式并不重要，只是两台机器上必须都要有它。

用 **scp** 把密钥复制到 *master* 和 *slave1* 上，或者用剪切加粘贴的办法也行。不要使用 **telnet** 或者 **ftp** 来复制密钥，甚至在内部网络上这样做也可能不安全。**scp** 是 SSH 软件的一部分。参考 20.11.3 节了解详情。密钥必须包含在两台机器的 **named.conf** 文件里。因为 **named.conf** 通常谁都有读权，而密钥不应该这样，所以要把密钥放在一个单独的文件里，让 **named.conf** 包含这个文件。例如，您可以把下面这小段代码

```
key master-slave1. {
    algorithm hmac-md5 ;
    secret "shared-key-you-generated" ;
} ;
```

放到文件 **master-slave1.tsig** 里。这个文件的模式是 600，它的属主应该是 **named** 的 UID。在 **named.conf** 文件中加入一行：

```
include "master-slave1.tsig" ;
```

这部分配置只是定义了密钥。要让它们能够实际用于签名和核实更新，主服务器需要有一条用于传送的密钥，而从服务器需要用 `server` 语句和 `keys` 资源标识出主服务器。例如，在主服务器上的 zone 语句中加上下面这行：

```
allow-transfer { key master-slave1. ;} ;
```

而把下面这行加到从服务器的 **named.conf** 文件里。

[19] 它不是真的随机数，甚至都不是伪随机数，它是对资源记录 DNSKEY 的散列值。

www.TopSage.com

```
server master's-IP-address { keys { master-slave1. ; } ; } ;
```

如果主服务器允许动态更新，那么它也能在 zone 语句的 allow-update 子句中使用密钥。

我们已经用通用名 master 和 slave1 来标识服务器和密钥。如果您在许多区上都使用了 TSIG，您应该也在命名方案里把区包括进来，帮助您把一切搞清楚。

要测试您的 TSIG 配置，可以运行 **named-checkconf** 来核对语法是否正确。然后用 dig 尝试从 slave1 和其他机器分别做一次区传送（**dig** *@master* **axfr**）。前一次区传送应该成功，而后一次应该失败，并返回诊断信息“Transfer failed.（传送失败）”。要彻底保证一切正确，可以删除 allow-transfer 子句，再用 **dig** 命令试试。这回应该两次区传送都成功。（不要忘记把 allow-transfer 再加回来！）最后做一次测试，增加主服务器上该区的序列号，观察从服务器上的日志文件，看它是否知道了变化，并且做了区传送。

当您第一次开始使用事务签名的时候，可以让 **named** 运行在调试级别 1 下（参考 15.16.1 节了解有关调试模式的知识）运行一段时间，看看有没有产生什么出错信息。BIND 比较老的版本不理解带签名的消息，并会发出抱怨，有时甚至达到了拒绝加载区数据的程度。

TKEY 是 BIND 9 让两台主机自动生成一个共享秘密的密钥，而不用打电话或安全拷贝来分发密钥的一种机制。它使用一种称为 Diffie-Hellman 密钥交换的算法，在这种算法中，每一方生成一个随机数，对它进行某种数学变换，并将结果发送给另一方。然后每一方将自己的数字与它接收到的数字进行数学变换来得到同一个密钥。窃密者可能会窃取到传输的加密数字，但是无法进行数学反推[20]。遗憾的是，让 TKEY 能投入实用，并且避免把 TSIG 密钥保存在配置文件里的代码尚未实现，不过还是有希望的，因为这件事情已经在任务表里安排了。

当主从服务器之间使用 TSIG 密钥和事务签名的时候，您应该让服务器的时钟与 NTP 保持同步。如果时钟过于不准（在 BIND 9 里是超过 5 分钟），那么就无法验证签名了。这个问题调试起来很难发现。参考 29.12 节了解有关 NTP 的更多知识。

目前还不可能用 TSIG 密钥来保护 BIND 和微软的服务器之间通信的安全。微软以一种非标准的方式（GSS-TSIG）来使用 TSIG，而且在没有签订保密协议的条件下则不能共享其方案的细节。各方面正在协商解决这一冲突。SIG(0)是另一种事务签名的机制，用在服务器之间，或者用在做动态更新的一方和主服务器之间。它使用公钥加密，参见 RFC 2535 和 2931 了解细节。

15.15.4 DNSSEC

DNSSEC 是一套对 DNS 的扩展，它使用公钥加密，对区数据的来源进行身份验证，并验证其完整性。也就是说，该扩展功能允许 DNS 客户端询问这样的问题：“该 DNS 数据是否真的来自于该区的所有者?”以及“这是否确实是那个区的所有者发送的数据？”。

DNSSEC 依赖于一条信任链：根服务器提供顶级域的有效信息，顶级域提供二级域的有效信息，以此类推。BIND 的配置项 trusted-keys 能让我们启动这个过程，在根和顶级域的安全得到保护之前，先保护部分 DNS 树的安全。

公钥加密系统使用两个密钥：一个加密（签名），另一个解密（验证）。发布者用一个秘密的“私有”密钥对他们的数据签名。任何人可以用广泛分发的与之匹配的“公共”密钥来验证签名的有效性。如果公钥能正确地对区文件解密，则这个区肯定已经用相应的私钥加过密。诀窍在于要保证您用于认证的公钥是可信的。公钥系统允许一个实体对另一个实体的公钥作签名，由此保证了密钥的合法性，

[20] 使用的数学方法称为离散对数问题，它依赖于一个事实，即对于模运算而言，做幂运算容易，但通过取对数来作幂的逆运算几乎是不可能的。

www.TopSage.com

因此才有了“信任链”一说。

DNS区中的数据太多，不能使用公钥加密法来加密——因为这样加密速度太慢。相反，因为数据不是保密的，所以对数据进行安全性的散列（hash，也音译为哈希）处理（例如MD5校验和）并用区的私钥对散列结果作签名。散列结果类似于数据的指纹，已签名的指纹称为数字签名。签名被追加到它们所验证的数据上，作为被签名的区文件的RRSIG记录。

要验证签名，就要使用签名者的公钥对它解密，对数据执行相同的安全散列算法，并把计算得到的散列值与解密得到的散列值进行比较。如果它们匹配，则您通过了对签名者的认证并验证了数据的完整性。

在DNSSEC系统中，每个区都有自身的公钥和私钥。私钥对每个RRset（资源记录集）（也就是说，用于同一主机的每一组同类型记录）进行签名。公钥验证签名并以一条DNSKEY资源记录的形式包含在区数据中。

父区对其子区的公钥进行签名。**named**通过把子区的DNSKEY记录对比父区的签名来核查以验证其真实性。为了验证父区密钥的真实性，**named**可以再检查父区的父区，并依此类推到根区。根区的公钥包含在根的线索文件中。

在我们深入讨论密钥生成和对区签名的机制之前，我们需要诚实地告诉读者DNSSEC的现状，以及它给系统管理员带来的影响。系统管理员可能需要在未来一两年里开始同DNSSEC打交道，但这肯定不会出现在“本周必做任务单”中。许多应用都吵着要一种公钥加密体系（public key infrastructure，PKI），DNS是提供PKI的主要候选服务。不过，我们碰到了一个有点儿死循环的问题。我们需要保证，在我们能将给其他Internet事务的密钥用于验证DNS是否可信之前，DNS是安全的。但是，我们又需要一个公钥加密体系来保证DNS是安全的。

最近在DNSSEC规范上的变化已经让它在向可以部署的目标上，比以前又接近了很多。原来的DNSSEC规范要在父域和子域都为子区保留签名密钥的拷贝，所以不管子域还是父域想要改变其密钥，都要求不停地进行通信。既然每个密钥都有多个拷贝，那么当两个拷贝不同步的时候，就不清楚应该相信哪一个了。

RFC 4034已经引入了一种新的资源记录类型（DS，代表delegation signer），将密钥分成key-signing密钥和zone-signing密钥，并且改变了一些资源记录的名字：

- KEY变为DNSKEY，用于密钥；
- SIG变为RRSIG，一个资源记录集（RRset）的签名；
- NXT变为NSEC，表示授权中的下一个安全项。

KEY和SIG仍然用于SIG(0)和TSIG。DNSKEY、RRSIG和NSEC用于DNSSEC，而NXT则被废弃。从SIG到RRSIG的变化，以及从NXT到NSEC的变化都不大，之所以改变名字是为了不会把老软件和新方案搞混。

在一个资源记录DNSKEY中包括的密钥可以是key-signing密钥（KSK），也可以是zone-signing密钥（ZSK）。有一个叫做SEP（代表secure entry point）的新标志对它们进行区分。这个标志字段的第15位被设为1代表KSK，设为0代表ZSK。这一约定让KSK按十进制算时为奇数，ZSK为偶数。

DS记录只出现在父区里，它指示一个子区是安全的（签过名的）。它还标识子区用于自行对自己的密钥RRset做签名的密钥。DS记录包括一个密钥标识符（一个5位的数字）、加密算法、摘要类型、以及允许（或者用于）对子区的密钥资源记录签名的公钥记录摘要。

如何改变父区和子区中现有密钥的问题是一个棘手的难题，似乎注定要求父区和区之间进行交流与协作。创建DS记录、使用key-signing密钥和zone-signing密钥，以及使用多个密钥对，都有助于解决这个问题。

DNSSEC可以生成和签发多个密钥，这样一来，就能做到从一个密钥到下一个密钥的平滑过渡。子区可以改变自己的zone-signing密钥，不必通知父区，它只需要在自己改变key-signing密钥的时候协调好父区即可。随着密钥的改变，会有一段时间新老密钥都有效，一旦Internet上的缓存值已经过期作废，老的密钥也就退出使用了。

www.TopSage.com

当前的 BIND 软件已经从发布版本中删除了 OpenSSL，所以如果您想要使用 DNSSEC，那要么找一个支持 DNSSEC 的 BIND 软件包，要么得去 www.openssl.org 下载 SSL 库，然后在重新编译 BIND 的时候打开对加密的支持（使用**./configure** 的**--with-openssl** 选项）。如果您没有这么做，那么 **dnssec-keygen** 会报错。不过，它仍然能生成 TSIG 密钥，因为这些密钥不需要 OpenSSL。

创建和使用有签名的区要做几步工作。首先，为这个区生成一个或者几个密钥对。例如：

```
# dnssec-keygen -a DSA -b 768 -n ZONE mydomain.com
```

使用 DSA 算法生成一个 768 位的密钥对，给一个叫做 mydomain.com 的区签名。加密算法根据密钥长度可以选择：RSAMD5（512~4096）、RSASHA1（512~4096）、DH（128~4096）和 DSA（512~1024 且能被 64 整除）。您可以带上**-f KSK** 标志（本例没有采用），设定它的 SEP 位，把生成的密钥确定为一个 kye-signing 密钥。**dnssec-keygen** 在标准输出上打印如下信息：

```
Kmydomain.com.+003+50302
```

此处的 Kmydomain.com 是密钥的名字，003 是 DSA 算法的标识符，50302 是密钥的标识符。**dnssec-keygen** 创建包含公钥和密钥的两个文件：

```
Kmydomain.com.+003+50302.key      # Public key
Kmydomain.com.+003+50302.private  # Private key
```

密钥（private key）给一个区的数据记录做签名，公钥用来核对签名。公钥插入到区文件的位置一般正好在 SOA 记录的后面。

在理想情况下，任何密钥对的密钥部分要离线保存，或者至少要放在一台没有连到 Internet 的机器上。这一预防措施对于动态更新的区不可能做到，而对于 zone-signing 密钥则不切实际，但对于 key-signing 密钥则完全合理，因为 key-signing 应该存在很长时间。

不看看 **dnssec-keygen** 生成的文件的样子，很难对它们有个感性认识，所以让我们瞥一眼吧：

Kmydomain.com.+003+50302.key

```
mydomain.com. IN DNSKEY 256 3 3
    BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1WU6D7O7/GiBXwxAsvpgH6
    sNXE3uwZVaQFxvDHfa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6zRBoO
    3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4yL8N6B32akBTiIMLp01FOJe
    xqLe6QrJVE21eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOsGRfOf4hy
    lOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTMVzTk
    qA4KJsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
    WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcK
    g2A3LHpmg1S59utmpxJa
```

Kmydomain.com.+003+50302.private

```
Private-key-format: v1.2
Algorithm: 3 (DSA)
Prime(p): tLZTYI1WU6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDHfa6amy3JSSilc
    RNfiiOs3LfoyZzUWOceVo6zRBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4y
    L8N6B32akB
Subprime(q): w5HLHyxHPoQnoRN9GPQ+Hi6toHU=
Base(g): TiIMLp01FOJexqLe6QrJVE21eXzRqC58TC25R6TPMoOH6cuue5w8eNphcs
    OsGRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTM
    VzTkqA4K
Private_value(x): GqcQz8K56CmUxgo6ERuyEWMLVME=
Public_value(y): JsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
    WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3L
    Hpmg1S59utmpxJa
```

www.TopSage.com

为了方便密钥换班，您应该生成几对 KSK 密钥，由父区对它们都做签名。一旦您和您的父区有了一种确定的信任关系（一个由父区签名的 KSK 被加到 DS 记录里，该记录授权您的区是安全的），您就可以用这种关系启动密钥换班过程。只要把那个密钥发给父区，确定您是谁，并带有想要让 DS 记录签名和指示的其他 KSK。在对新密钥签名之后，原来的密钥就退出使用了。

一旦父区对您的 KSK 做了签名，并且把它插入到您的 DS 记录中，就做好了对区的实际数据进行签名的准备。签名操作要把一个正常的区数据文件作为输入，并且在每一组资源记录之后紧接着加上 RRIS 和 NSEC 记录。RRSIG 记录是实际的签名，而 NSEC 记录支持否定回答的签名。

用 **dnssec-keygen** 命令给一个区做签名。例如，下面的命令：

```
# dnssec-signzone -o mydomain.com db.mydomain
   Kmydomain.com+003+50302
```

读取区文件 **db.mydomain**，生成这个区文件的签名后版本，叫做 **db.mydomain.signed**。这条命令还创建了两个密钥集文件：

```
keyset-mydomain.com.      # Keyset
dsset-mydomain.com.       # Secure delegation keyset
```

这两个密钥集文件分别用于对区和密钥做签名。它们的内容如下：

keyset-mydomain.com.

```
$ORIGIN .
mydomain.com   172800  IN  DNSKEY   256 3 3 (
                           BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1W
                           U6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDH
                           fa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6z
                           RBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx
                           4yL8N6B32akBTiIMLp01FOJexqLe6QrJVE21
                           eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOs
                           GRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9
                           hjzskX64e0QeeENXRV8sFvTMVzTkqA4KJsBC
                           clVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn
                           0RO0mpH6ZUIlWaSIE1pAxaZ7ynD4hT1RB5br
                           2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3
                           LHpmg1S59utmpxJa
                           ) ; key id = 50302
```

dsset-mydomain.com.

```
mydomain.com. IN DS 50302 3 1 1B44471AFD5B4F4463BB3A0D7B66B6ABC018DA96
```

在当前目录下，任何用于子区的密钥集文件，也都被签名，并且加入到 DS 记录里，这个 DS 记录把若干个安全的子区授权给这个子区的服务器，并且用签过名的密钥验证它们的身份。对一个区做签名要花很长一段时间。

如果把一个带签名的区作为参数传给 **dnssec-signzone** 命令，那么濒临作废的任何记录的签名都会续用。“濒临作废”的定义为时间已经走完了有效期的四分之三。在默认情况下，签名 30 天有效，但在 **dnssec-signzone** 命令行上可以指定一个不同的有效期。重新签名一般会引起变化，但不会自动修改序列号，所以系统管理员需要手工更新序列号。区的从服务器必须执行一次区传送操作来让它们重新保持同步。

下面的例子给出了区数据在签名前后的变化[21]。先是签名之前：

[21] 我们试图通过只用两台主机及其 A 和 MX 记录来简化这个例子，但是 **dnssec-sigzone** 要求一个“真实的”区，所以必须加上 SOA、NS 和 DNSKEY 才能构成一个最简单的例子。不好意思，弄得这么乱。

www.TopSage.com

```
$TTL 172800 ; 2 days
@               IN  SOA     ns.mydomain.com. hostmaster.mydomain.com. (
                                2006081200      ; serial
                                7200            ; refresh (2 hours)
                                3600            ; retry (1 hour)
                                1728000         ; expire (2 weeks 6 days)
                                172800          ; minimum (2 days)
                                )

                IN  NS      ns.cs.colorado.edu.

mydomain.com. IN  DNSKEY 256 3 3
    BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1WU6D7O7/GiBXwxAsvpgH6
    sNXE3uwZVaQFxvDHfa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6zRBoO
    3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4yL8N6B32akBTilMLp01FOJe
    lOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTMVzTk
    qA4KJsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
    WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcK
    g2A3LHpmg1S59utmpxJa

anchor          IN  A       128.138.242.1
                IN  A       128.138.243.140
                IN  MX      10 anchor
                IN  MX      99 @
awesome         IN  A       128.138.236.20
...
```

签名之后为：

```
; File written on Thu Nov 11 17:41:25 2006
; dnssec_signzone version 9.3.0
mydomain.com. 172800 IN SOA ns.mydomain.com. hostmaster.mydomain.com. (
                                2006081200  ; serial
                                7200        ; refresh (2 hours)
                                3600        ; retry (1 hour)
                                1728000     ; expire (2 weeks 6 days)
                                172800      ; minimum (2 days)
                                )
        172800 RRSIG    SOA 3 2 172800 20061211164125 (
                          20061111164125 50302 mydomain.com.
                          BElneYxZ3g9JnKbXdnmPhKVWfd13JTU8ajOO
                          5dQta2WeBAatNuWt8dQ= )
        172800 NS       ns.cs.colorado.edu.
        172800 RRSIG    NS 3 2 172800 20061211164125 (
                          20061111164125 50302 mydomain.com.
                          BLG6LRrXtRHRdRFtTOmlQsadOIefqHAq5Rid
                          PHZ74vOl/UkEW6wY6VA= )
        172800 NSEC     anchor.mydomain.com. NS SOA RRSIG NSEC DNSKEY
        172800 RRSIG    NSEC 3 2 172800 20061211164125 (
                          20061111164125 50302 mydomain.com.
                          BCz31GPChdQrmNrZypv4xxmXDCThZ0IlkEGL
                          TSkf7Q+TmCDmAADxmBE= )
        172800 DNSKEY   256 3 3 (
                          BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1W
                          U6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDH
                          fa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6z
                          RBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx
                          4yL8N6B32akBTilMLp01FOJexqLe6QrJVE21
                          eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOs
```

www.TopSage.com

```
                                GRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9
                                hjzskX64e0QeeENXRV8sFvTMVzTkqA4KJsBC
                                clVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn
                                0RO0mpH6ZUIlWaSIE1pAxaZ7ynD4hT1RB5br
                                2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3
                                LHpmg1S59utmpxJa
                                ) ; key id = 50302
        172800  RRSIG     DNSKEY 3 2 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BAgZDfk/YCOhVfuoyG5pgfyFCmsGqg4W7uuM
                                Rm5eNP9Bn0EbBnuT6X0= )
anchor.mydomain.com. 172800 IN A 128.138.242.1
        172800  IN A       128.138.243.140
        172800  RRSIG     A 3 3 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BIRtKW0Um7ItfbPqRew+jKo152WJh+4nHkmK
                                1ePNxjsQWcgaKm5jiMU= )
        172800  MX        10 anchor.mydomain.com.
        172800  MX        99 mydomain.com.
        172800  RRSIG     MX 3 3 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BGtmN2u30y1pMDzstWGgWZfXB3lDlmy5W6DP
                                t/8D31QpyYNBjJPb8J4= )
        172800  NSEC      awesome.mydomain.com. A MX RRSIG NSEC
        172800  RRSIG     NSEC 3 3 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BArN6oES72gzFgQmBHL3NzlquMbDbLfpvj7J
                                3CSb/c8U/bciWGXsV3Q= )
awesome.mydomain.com. 172800 IN A 128.138.236.20
        172800  RRSIG     A 3 3 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BJ/qWBgLgS/2N5CoXGnI4vs91SsyIBKKfoq9
                                R+VsMpRmnVrSi1DU1n8= )
        172800  NSEC      mydomain.com. A RRSIG NSEC
        172800  RRSIG     NSEC 3 3 172800 20061211164125 (
                                20061111164125 50302 mydomain.com.
                                BKoByqF5wUceb2vc8H2uealgKrejH4VZ0S5m
                                Q4KukWCUo2IAFX+msQ4= )
```

由此可见，有了签名的区真难看。它们一般比原来的区要大 4~10 倍，而且原本良好的逻辑次序也没了。实际而言，带签名的区文件不再适合常人直接阅读，而且也不能手工编辑，因为有了 RRSIG 和 NSEC 记录。里面没有用户自己能处理的部分了！

RRSIG 记录包含丰富的信息：

- 带签名的记录集合的类型；
- 使用的签名算法（在我们的例子中，它是 3，DSA 算法）；
- 带签名的记录集的 TTL；
- 签名过期的时间（形如 *yyyymmddhhssss*）；
- 对记录集签名的时间（也是形如 *yyyymmddhhssss*）；
- 密钥标识符（在我们的例子中是 50302）；
- 签名者的名字（mydomain.com.）；
- 最后是数字签名本身。

要使用带签名的区，就要更改 mydomain.com 域 **named.conf** 文件里 `zone` 语句中的 `file` 参数，让它指向 **db.mydomain.signed**，而不是 **db.mydomain**。就该这样！

www.TopSage.com

15.15.5 否定回答

数字签名适用于正面回答，例如“这是主机 anchor.cs.colorado.edu 的 IP 地址，附有签名，证明它的确来自于 cs.colorado.edu 并且数据是有效的。”但是类似“没有这样的主机”的否定答复又如何呢？这样的否定响应通常不返回任何可签名的记录。

在 DNSSEC 中，这个问题用 NSEC 记录来处理，它按照规范的排列顺序列出了区的下一条记录[22]。如果在 cs.colorado.edu 中 anchor 后的下一条记录是 awesome.cs.colorado.edu，那么在有对 anthill.cs.colorado.edu 的查询到达时，响应将会是一条带签名的 NSEC 记录，例如：

```
anchor.cs.colorado.edu. IN NSEC awesome.cs.colorado.edu A MX NSEC
```

这条记录说明在 cs.colorado.edu 区中紧跟 anchor 之后的名字是 awesome.cs.colorado.edu，anchor 至少有一条 A 记录、MX 记录和 NSEC 记录。如果主机存在，但要查询的记录类型不存在，也返回 NSEC。例如，如果查询 anchor 的 LOC 记录，则返回 anchor 的同一个 NSEC 记录并显示 anchor 只有 A、MX 和 NSEC 记录。

一个区中的最后一条 NSEC 记录又折回到该区的第一个主机。例如，zamboni.cs.colorado.edu 的 NXT 记录将往回指向第一条记录，即 cs.colorado.edu 自身的记录：

```
zamboni.cs.colorado.edu. IN NSEC cs.colorado.edu A MX NSEC
```

NSEC 记录提供了一条遍历区的简单途径，它向持续存在的查询方提供完整的区传送，一次一条记录。这对于安全/保密来说不太好。

15.15.6 微软和 DNS

Window 使用 SRV 资源记录来查找所有信息：名字服务器、打印机、文件系统等。微软在 SRV 记录的实现中遵循 IETF 的规范，但是它通过使用一种安全的动态更新在 DNS 中插入记录的方法是非标准的。微软使用一种基于共享秘密的事务签名的变体，叫做 GSS-TSIG。共享秘密的密钥是通过 Kerberos 从 Kerberos KDC（Key Distribution Center，密钥分发中心）获得的。目前，微软的实现用到了厂商扩展字段，因此与 Kerberos 5 的开放源代码版本不兼容。

如果您希望运行 Windows 并使用 SRV 记录，就不得不清除现有的 Kerberos 环境并在您的网络上运行 Windows Kerberos 服务器。对于一些大量使用 Kerberos 基础设施的网点来说，能解决这个问题可算是一项奇迹。微软在用的协议，其开放程度正好够拿到公司的采购单，但不足以让其他任何人与之互操作，加入他们的市场。让我们指望不需要目前的保密协议，微软也能提供有关它们所做扩展的文档说明，这样 ISC 就能让它的 TSIG 同微软的 GSS-TSIG 实现互操作。

发布 Windows 2000 之后一个星期左右，DNS 根服务器上的查询负载激增。稍作分析便发现是配置错误的 Windows 2000 系统正在尝试动态地更新根区或顶级区。结果对根服务器 A 的 UDP 查询数量增加了两倍以上。使事情变得更糟的是，当它们的更新请求遭到拒绝时，Windows 2000 系统启动一个 TCP 连接请求一个 KEY 记录，并尝试进行一次有身份验证的动态更新。根服务器 A 没有时间处理最终造成的不计其数的 TCP 连接请求。

影响到 Internet 基础设施的 bug 不但严重，而且有可能引起尴尬，所以我们装作认为到目前为止

[22] 这种排序方式是根据字母表的顺序，但是先列出 DNS 树的较高层域名。例如，在 cs.colorado.edu 区中，cs.colorado.edu 排在任何 host.cs.colorado.edu 之前。在层次结构的同一层次中，根据字母表的顺序排序。

www.TopSage.com

微软已经纠正了这个问题。我们希望把这一节变成一个历史脚注，只是为了全面起见，我们复查了新安装的 Windows XP SP2 系统的网络流量。令人惊异的是，我们发现问题依然故我般地存在。

这个问题最常见（但不全是）的地方是 RFC 1918 定义的私有地址的反向区。有关这些地址的信息绝不应该跑出本地环境，但是 Windows 机器却试图更新它们自己所在的域，而且如果失败了，它们就继续沿着 DNS 名字树向上，直到到达根服务器为止。

例如，如果 DHCP 服务器给 Windows 机器分配了 192.168.1.10 这个地址，它就会尝试动态更新 1.168.192.in-addr.arpa 这个域。如果失败，就尝试更新 168.192.in-addr.apra，然后再试 192.in-addr.apra，再试 in-addr.arpa，最后试 arpa。

没有重设其 DNS 默认配置的 Windows 系统就要为一种连续的、大规模的、缓慢的、无意中的、分布式的对根名字服务器的拒绝服务攻击做贡献。要让 Windows 2000 或者 XP 系统不去尝试更新根区，并且让它们和您的 UNIX 或者 Linux 名字服务器更好地合作，就要试试以下步骤：

- 鼠标右键点击“My Network Places（网上邻居）”，选择“Properties（属性）”，显示出叫做“Network Connections（网络连接）”的窗口；
- 依次用右键点击每条连接，选择“Properties（属性）”；
- 点击“Internet Protocol (TCP/IP)”，然后点击“Properties（属性）”按钮；
- 点击属性页底部的“Advanced...（高级...）”按钮；
- 点击顶部的“DNS”页；
- 到下面去掉“Register this connection's address in DNS（在 DNS 中注册这个连接的地址）”旁边的勾；
- 一路点击“OK（确定）”退出。

为了争取时间，根区的操作人员把 RFC 1918 地址空间的反向区授权给称为 prisoner.iana.org、blackhole1.iana.org 和 blackhole2.iana.org 的几台服务器。这几台服务器是这些区的权威服务器（prisoner 是主服务器，两台 blackhole 是从服务器），它们截获由 Windows 系统产生的动态更新，这些 Windows 系统想要告诉世界，“mypc”这个名字应该关联到 10.0.0.1 这个地址上。这几台 iana.org 服务器采用 anycast，在大多数大型 ISP 那里都有实例，所以从本地网点跑出来的假的 DNS 更新消息不会把根服务器上对 Internet 的认识完全搞糟。

且慢，这还不是微软 DNS 恶习的终结。如果它们的首选 DNS 服务器没有在一秒内响应一次查询，Windows 系统会迅速紧张起来，同时向它们知道的每个 DNS 服务器发出查询请求[23]。换句话说，一遇到服务器过载或者网络拥塞的情况，Windows 就会尝试产生尽可能多的网络和服务器流量。您可以通过禁用 DNS Client（DNS 客户机）服务来关闭这种做法，但要注意，这样做了也会禁用每台机器的本地 DNS 缓存。下面是禁用 DNS Client 服务的步骤：

- 从 Start（开始）菜单选择 Settings（设置）->Control Panel（控制面板）；
- 在控制面板中，双击 Administrative Tools（管理工具）；
- 点击“Services and Applications（服务和应用程序）”左边的“+”，展开它；
- 选择左边的 Services（服务），双击右边的 DNS Client（DNS 客户机）；
- 点击“Startup type（启动类型）”旁边的下拉菜单，选择 Manual（手动）；
- 点击 Stop（停止）按钮，停止该服务已在运行的副本；
- 一路点击“OK（确定）”退出。

如果您没有在用微软的 Active Directory 服务，那么您或许最好使用 UNIX/Linux 的 DNS 服务器，而避免使用微软的 DNS 实现。遗憾的是，Active Directory 让事情复杂了，但您可以用 BIND 支持 Active Directory。

[23] 这个做法显然是故意的。参考在微软知识库文章 286834（support.microsoft.com）中援引的微软 DNS 白皮书了解更多的信息。

www.TopSage.com

最后一点：微软的服务器在查询一个区，而这个区的服务器都是残缺服务器的时候，就会出现一场混乱。如果您的站点授权给了似乎是残缺的服务器，那么比较好的做法是，至少安装一个空的区文件，替代那些缺少的服务器。这项措施会防止查询残缺名字的微软服务器发生混乱，冲击根服务器和gTLD服务器。

15.16 测试和调试

named 提供几种内置的辅助调试手段，其中最重要的是可配置性非常灵活的日志功能。您可以在命令行指定调试级别或者用 **rndc** 设置它们。您还可以命令 **named** 把运行统计结果转储到一个文件，并用 **dig** 这样的外部工具验证域名查询。

15.16.1 日志功能

named 的日志功能的灵活性足已让您震惊。BIND 原本使用系统日志来报告错误消息和异常情况。最近的BIND版本通过再多加一个间接层，并支持直接将日志写入文件，推广了系统日志的概念。有关系统日志的更多信息，请参考第10章。在深入说明之前，让我们看看BIND日志术语的小词汇表，如表15.12所示。

表 15.12 BIND 的日志术语表

术 语	含 义
channel（通道）	消息能去的地方：syslog、文件或者/dev/null[a]
category（类别）	named 能够产生的消息类型。例如，有关动态更新的消息或者有关答复查询的消息
module（模块）	产生消息的来源模块的名字
facility（设备）	syslog 设备名。DNS 没有它自己特有的设备，但是您可以选择所有的标准设备
severity（严重性）	出错消息的“糟糕程度”。syslog 称之为级别

a. /dev/null 是一个伪设备，它丢弃所有的输入。

BIND日志是在 **named.conf** 中用 logging 语句配置的。首先定义通道，即消息可能有的目的地。然后，让几种不同类别的消息去往特定的通道。

当产生一条消息时，在其发源地就给它指派了一种类别、一个模块和一种严重性级别。然后它被分发到与其类别和模块相关的所有通道中。每个通道有一个严重性过滤器，用来指定一条消息要通过过滤器所必须具有的严重性级别。通向系统日志的通道也根据**/etc/syslog.conf** 中的规则进行过滤。

这里是 `logging` 语句的结构：

```
logging {
    channel_def;
    channel_def;
    ...
    category category_name {
        channel_name;
        channel_name;
        ...
    };
};
```

一、通道

根据通道是文件通道或系统日志通道，*channel_def*（通道定义）略显不同。您必须为每个通道选

www.TopSage.com

择是 file 或是 syslog，一个通道不能同时二者都是。

```
channel channel_name {

    file path [versions numvers | unlimited] [size sizespec];
    syslog facility;
    severity severity;

    print-category yes | no;
    print-severity yes | no;
    print-time yes | no;
};
```

对于一个文件通道来说，*numvers* 说明要保存文件的多少个备份版本，*sizespec* 指定在轮换文件之前，它最大应该增长到多大（例如：2048、100k、20m、15g、unlimited、default）。如果您给一个文件通道起了名字叫 **mylog**，那么这个文件的轮换版本就叫做 **mylog.0**、**mylog.1**，以此类推。

在系统日志的情况下，*facility* 指定用来对消息作日志记录的设备名称。它可以是任何标准设备。实际上，只有 daemon 以及 local0 到 local7 是合理的选择。有关系统日志设备名称的列表，请参考 10.4.3 节。

channel_def 中的其他语句是可选的。*severity* 可以使用的值为（按降序）critical、error、warning、notice、info 或 debug（带有一个可选的数字级别，例如 severity debug 3）。dynamic 也是可以识别的值，它与服务器的当前调试级别相匹配。

不同的 print 选项增加或删除消息前缀。系统日志预先在写入日志的每条消息前设置时间和报告主机，但没有严重性或类别。产生消息的源文件名（模块）也可作是一个 print 的选项。启用 print-time 仅对文件通道有意义，系统日志将按自己的时间作日志记录。

表 15.13 中列出了默认情况下预先定义的 4 个通道。默认通道对于大多数安装应该是没有问题的。

表 15.13 BIND 中预先定义的日志通道

通道名称	执行内容
default_syslog	发送到 syslog、设备为 daemon、严重性为 info
default_debug	日志记录到文件 **named.run** 中，严重性设置为 dynamic
default_stderr	发送到 **named** 进程的标准出错设备（stderr），严重性为 info
null	丢弃所有的消息

二、类别

类别由程序员在编写代码的时候确定，他们按照主题或者功能而不是严重性来组织日志消息。表 15.14 给出了当前的消息分类。

表 15.14 BIND 的日志类别

类 别	包括的内容
client	客户机请求
config	配置文件分析和处理
database	有关数据库操作的消息
default	没有明确分配通道的类别
delegaton-only	按 delegation-only 区对 NXDOMAIN 的查询
dispatch	把传入的包分发给多个服务器模块
dnssec	DNSSEC 消息

www.TopSage.com

大家网 TopSage.com

续表

类　别	包括的内容
general	未分类的消息
lame-servers	应该是服务一个区的服务器，但却没能这样[a]
network	网络操作
notify	有关“区已更改”通知协议的消息
queries	服务器接收到的每个查询的简短日志消息（！）
resolver	DNS 解析，例如，客户机的递归查询
security	批准/未批准的请求
unmatched	**named** 不能对查询进行分类（错误的类，没有 view）
update	有关动态更新的消息
xfer-in	服务器正在接收的区传送
xfer-out	服务器正在发送的区传送

a. 父区和子区都可能出问题，不作调查无法区分。

三、日志消息

BIND 9 的默认配置是：

```
logging {
    category default { default_syslog; default_debug; };
};
```

对 BIND 做重大更改时，应该监视日志文件，可能还要提高日志级别。一旦 **named** 稳定后，就重新进行配置，仅保留严重的消息。

从域名查询的日志记录里能学到相当多的东西。您可以验证 `allow` 子句在起作用、看看是谁在查询您、确定出有问题的客户机等。在对配置进行了重大调整之后，对查询做日志是检查效果的良好方法，如果您已经知道调整前查询负载通常的情况，那么这样做就格外有用处了。

要启动对域名查询的日志记录，只要把 `queries` 类别指向一个通道就可以了。把日志写入 syslog 没有直接写入文件效率高，所以在您要对每次查询做日志时，可以在本地磁盘上使用一个文件通道。要准备大量的磁盘空间，一旦您得到了足够多的数据，就要迅速关闭对域名查询做日志的设置。（**rndc querylog** 可以打开和关闭这一功能。）

view（视图）调试起来比较麻烦，但还好，匹配一次查询的 view 也和查询本身一起被记入日志。

以下列出了一些常见的日志消息。

- *Lame server*（残缺服务器）。如果您得到的这条消息是有关您自己的一个区，那么是您把某些东西配置错了。如果这条消息是有关外界 Internet 上的某个区的，那么它相对来说无害，它是其他人的问题。丢掉这类消息的好做法是把它指向一个 null（空）通道。
- *Bad referral*（无效推荐）。这条消息表明在区的名字服务器之间有一次误通信。
- *Not authoritative for*（非权威性）。从服务器无法获取对一个区的权威数据。可能它指向了错误的主服务器，或者可能是主服务器装载该区时出了问题。
- *Rejected zone*（被拒绝区域）。因为区文件包含错误，所以 **named** 拒绝它。
- *No NS RRs found*（找不到 NS RR）。区文件在 SOA 记录之后没有 NS 记录。可能是遗漏了这一记录，或者是它们没有以一个制表符或其他空白开头。在后一种情况中，记录没有附加在

www.TopSage.com

SOA 记录所在的区上，因此解释有误。

- *No default TTL set*（没有默认的 TTL 设置）。设置默认 TTL 的首选途径是在区文件的开头使用 $TTL 子句。这一错误消息表明$TTL 子句丢失。在 BIND 9 中，必须使用$TTL。
- *No root name server for class*（没有类的根名字服务器）。您的服务器无法找到根名字服务器。检查线索文件，以及服务器与 Internet 的连接。
- *Address already in use*（地址已经在使用）。希望运行 **named** 的端口已经被另一个进程占用，可能是 **named** 的另一个副本。如果没看到另一个 **named** 在运行，那它可能已经崩溃并留下一个打开的 **rndc** 控制套接口，您不得不要找出并删掉它[24]。解决它的一个好办法是关闭它并重新启动 **named** 进程。

```
# /etc/init.d/named stop (或者在 Red Hat 里是 /sbin/service named stop)
# /etc/init.d/named start (或者在 Red Hat 里是 /sbin/service named start)
```

- *Dropping source port zero packet from ...*（丢弃从……来的源端口为 0 的包）。最近的 BIND 版本可以让您设定域名查询的源端口号，系统管理员可以使用这一功能向他们的防火墙加入规则，这些防火墙能够根据源端口识别出他们的 DNS 包。不过，0 不是合法的 TCP/UDP 端口号。如果出错消息和您的某一台主机有关，您应该改变那台机器上 **named.conf** 文件里的 query-source 指令，来修正这个错误。
- *Denied update from [...] for ...*（拒绝从[……]来的对于……的更新）。对一个区进行动态更新的尝试被拒绝，因为这个区的 **named.conf** 文件中有 allow-update 或者 update-policy 子句。

15.16.2 日志配置样例

下面是从 ISC 的 **named.conf** 中截取的片段，它用于一个繁忙的 TLD 服务器，展示了一个全面的日志方式。

```
logging {

    channel default_log {
        file "log/named.log" versions 3 size 10m;
        print-time yes;
        print-category yes;
        print-severity  yes;
        severity info;
    };

    category default { default_log; default_debug; };

    channel xfer-log {
        file "log/xfer.log" versions 3 size 10m;
        print-category yes;
        print-severity yes;
        print-time yes;
        severity info;
    };
    channel db-log {
        file "log/db.log" versions 3 size 1M;
        severity debug 1;
        print-severity yes;
        print-time yes;
```

[24] 在 gTLD 服务器上，该消息可能说明 com 仍在载入中。

www.TopSage.com

```
        };

        category database { db-log; };
        category dnssec { xfer-log; };
        category xfer-in { xfer-log; };
        category xfer-out { xfer-log; };
        category notify { xfer-log; };
};
```

15.16.3 调试级别

named 的调试级别用 0~100 的整数表示。数字越大，表示输出信息越详细。0 级关闭调试，1 级和 2 级适用于调试您的配置和数据库，大于 4 的级别适合代码的维护人员使用。

您可以在 **named** 命令行用**-d** 标志调用调试功能，例如：

```
# named -d2
```

将以调试级别 2 启动 **named**。默认情况下，调试信息写入当前工作目录下的 **named.run** 文件，**named** 就是从这个目录启动的。**named.run** 文件增长速度很快，所以在调试过程中不要分神去忙别的事，否则在您回来后，将会遇到很严重的问题。

您还可以在 **named** 的运行过程中用 **rndc trace** 打开调试，这条命令将调试级别增加 1，或者用 **rndc trace** *level* 把调试级别设为指定的值。**rndc notrace** 把调试完全关闭。您还可以通过定义一个包括严重性规定的日志通道来启用调试，例如：

```
severity debug 3;
```

将级别 3 以下的所有调试消息发送到特定通道中。通道定义中的其他行指定这些调试消息出现的目的地。严重性级别越高，日志记录的信息越多。

监视日志或调试输出都能展示出错误配置 DNS 数据的频繁程度。域名尾部讨厌的点（或者说，缺少这个点）引发的 DNS 流量达到了引起人们警觉的程度。理论上讲，每个完整的域名后面都要求有这个点。

15.16.4 用 rndc 调试

表 15.15 列出了 **rndc** 接受的一些选项。不带参数执行 **rndc** 会列出一份可用命令的清单。产生文件的命令把文件放到 **named.conf** 中指定的 **named** 主目录中。

表 15.15 rndc 命令[a]

命　令	功　能
dumpdb	把 DNS 数据库转储到 **named_dump.db**
flush [*view*]	冲洗所有的缓存，或者冲洗一个特定 *view* 的缓存
flushname *name* [*view*]	把指定的 *name* 从服务器的缓存里冲洗掉
freeze *zone* [*class* [*view*]]	挂起对一个动态 *zone* 的更新
thaw *zone* [*class* [*view*]]	恢复对一个动态 *zone* 的更新
halt	中止 **named**，不保存没做完的更新
querylog	开关对传入查询的日志功能
notrace	关闭调试

www.TopSage.com

续表

命 令	功 能
reconfig	重新载入配置文件和任何新的区
refresh *zone* [*class* [*view*]]	安排维护 *zone* 的时间
reload	重新载入 **named.conf** 和区文件
reload *zone* [*class* [*view*]]	只重新载入指定的 *zone* 或者 *view*
restart[b]	重新启动服务器
retransfer *zone* [*class* [*view*]]	重新从主服务器拷贝 *zone* 的数据
stats	把统计信息转储到 **named.stats**
status	显示当前运行的 **named** 的状态
stop	保存没做完的更新，停止 **named** 的运行
trace	将调试级别加 1
trace *level*	改变调试级别

a. 这里的 *class* 参数和资源记录的一样，一般用 IN 代表 Internet。
b. 在 BIND 9（9.3.0）里尚未实现，但是承诺会实现。

rndc reload 让 **named** 重新读取自己的配置文件并重新装载区文件。在只更改了一个区并且您不希望重新装载所有区时，**reload** *zone* 命令就很方便，特别对于繁忙的服务器更是如此。您可以指定一个 *class* 和 *view* 只重新装载从区数据中选出的 *view*。**rndc cnfig** 命令可以很容易实现，不用做一次完整的重新装载，就能加入新的区，它重新读取配置文件，不用干扰现有的区，就能载入任何新区。

rndc freeze *zone* 停止动态更新，让挂起更新的日志文件和数据文件一致。在冻结该区之后，您可以手工编辑该区的数据。只要区被冻结了，那么就会拒绝动态更新。一旦您编辑完数据，就可以使用 **rndc thaw** *zone* 命令再次开始接受动态更新。

rndc dumpdb 让 **named** 把它的数据库转储到 **named_dump.db** 中。转储文件很大，不仅包括本地数据，还包括名字服务器上积累的所有缓存数据。我们的主名字服务器 colorado.edu 上的数据库缓存最近一次的转储超过了 16MB，但装载的区数据不到 200K。那里面包含了许多缓存的内容。

如果您正在一个 **chroot** 后的环境下运行 **named**，那么您必须提供给 **rndc** 到 **rndc** 套接口的路径，因为它不会在**/var/run** 里，而 **rndc** 则以为会在那里找到它。

使用类似下面的命令：

```
$ sudo rndc -l /var/named/var/run/rndc
```

在 BIND 9 中，确保 **named** 的版本和 **rndc** 的版本相匹配是很重要的，否则您会得到一则有关协议版本不匹配的出错消息。在升级的时候，只要安装它们两个就行。

15.16.5 BIND 统计

named 有一些汇总信息，当它接到 **rndc** 命令的要求时，可以把这些信息转储到其工作目录下的文件 **named.stats** 里：

```
$ sudo rndc stats
```

下面是从 isc.org 的一台繁忙的服务器上输出的例子：

www.TopSage.com

```
+++ Statistics Dump +++ (1088897026)
success 19917007
referral 981446459
nxrrset 9958824
nxdomain 199644113
recursion 0
failure 322556
--- Statistics Dump --- (1088897026)
```

这份统计显示出查询成功和失败的对比情况，而且把各种错误进行了分类。这台服务器成功地回答了1990万次查询、9.8亿次推荐、大约2亿次回答“no such domain（无此域）”（nxdomain）、大约1千万次回答“no such resource record set（无此资源记录集）”（nxrrset）、以及有32万次没有回答的情况。

如果这2亿次nxdomain错误（大于占10亿次查询的20%）的主要来源是因为用户输入错误和拼写错误的话，那么这个数字显得太多了。在这里的情况下，大多数错误源于Windows的错误配置和微软的解析器里的一个bug[25]，它在发出许多对A（地址）记录的查询时，没有按照DNS规范要求的那样，在数据字段里填入主机名，而是填入了一个IP地址。这些奇怪的查询显然含有一个类似于56这样的顶级域（或者IP地址串最后一个字节带的任何值），于是它们让本地的DNS服务器急急忙忙地奔向根服务器反复询问这个不存在的域，最终就产生了一条nxdomain错。

failure项计算了nxdomain（无此域）和nxrrest（无此资源记录集）之外的所有错误。

15.16.6 用 dig 调试

从shell可以使用3种工具来查询DNS数据库：**nslookup**、**dig**和**host**。在BIND的软件发布中3种工具都有。**nslookup**是这些工具中最古老的，而且总是随同BIND一起发布。**dig**代表domain information groper（域信息的探索程序），最初由Steve Hotz编写，后来Michael Sawyer针对BIND 9又重写了**dig**，它也和BIND一起发布。**host**由Eric Wassenaar编写，是另一个开放源代码的工具，其特点是输出对用户很友好，而且有检查区文件语法的功能。

我们推荐**dig**要胜过**nslookup**，**host**也不错。我们在这里只讨论**dig**。有时您可能会从这些工具得到不同的结果，这是因为它们使用的解析器库不同：**dig**和**host**使用BIND的解析器，而**nslookup**有它自己的解析器。

在默认情况下，**dig**使用本地的名字服务器，并且返回地址信息。而*@nameserver*参数则让它查询一台特定的服务器[26]。查询一台专门的服务器能让您检查确定您对一个区所做的任何改动是否已经传播到次级服务器和外界。如果您使用view，并且需要核实它们的配置是否正确，那么这个特别有用。

如果您指定了一个记录类型，那么**dig**将只查询这种类型记录的信息。伪类型ANY返回所有和指定名字有关的数据。另一个有用的标志是**-x**，它把一个IP地址的字节倒过来，做一次反向查询。**+trace**标志显示了从根服务器向下解析过程中的各个步骤。

dig的输出结果与下面的类似：

```
$ dig yahoo.com

; <<>> DiG 9.3.0rc2 <<>> yahoo.com
;; global options:  printcmd
;; Got answer:
```

[25] 微软已经在Windows 2000的Service Pack2中修正了这个缺陷。

[26] 如果该服务器不存在，那么老版本的**dig**会悄悄再去查本地服务器，所以要仔细看您得到的输出，核实数据的来源。从BIND 9.2起，在这种情况下，**dig**返回一条出错信息，不给出答案。

www.TopSage.com

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16507
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;yahoo.com.                    IN      A

;; ANSWER SECTION:
yahoo.com.          300     IN      A       66.94.234.13
yahoo.com.          300     IN      A       216.109.127.28
yahoo.com.          300     IN      A       216.109.127.29

;; AUTHORITY SECTION:
yahoo.com.          74804   IN      NS      ns3.yahoo.com.
yahoo.com.          74804   IN      NS      ns4.yahoo.com.
yahoo.com.          74804   IN      NS      ns1.yahoo.com.
yahoo.com.          74804   IN      NS      ns2.yahoo.com.

;; ADDITIONAL SECTION:
ns1.yahoo.com.      72593   IN      A       66.218.71.63
ns2.yahoo.com.      72596   IN      A       66.163.169.170
ns3.yahoo.com.      72594   IN      A       217.12.4.104
ns4.yahoo.com.      72594   IN      A       63.250.206.138
;; Query time: 6 msec
;; SERVER: 204.152.184.109#53(204.152.184.109)
;; WHEN: Sun Jul  9 16:03:43 2006
;; MSG SIZE  rcvd: 245
```

如果结果是权威结果，那么 **dig** 就在其 `flags` 列里标一个 `aa`。这里我们是向本地名字服务器查询 yahoo.com，所以回答不是权威的结果。

dig 的输出不仅包括域名信息，而且还包括发出的查询次数和回答的往返时间。输出的格式正好和一个区文件里使用的格式一样，当您为了获得自己的线索文件而查询根服务器的时候，这样的结果就特别方便使用。分号是一个区文件里的注释符，**dig** 用它在输出中插入注释。

当测试一个新配置的时候，要确保您从本地和远程主机都查到了数据。如果您能够用 IP 地址访问一台主机，但却不能用名字访问到它的时候，DNS 可能就是罪魁祸首。

15.16.7 残缺授权

在申请一个域名的时候，就是在请求将一部分 DNS 命名树授权给主名字服务器和 DNS 管理员。如果您从未使用这个域，或者更改了名字服务器但没有更新父域的粘合记录，则会导致“残缺授权”。

残缺授权可能造成很坏的影响。如果某个用户尝试联络残缺域中的一台主机，那么您的名字服务器将拒绝该查询。DNS 将重复尝试查询几百次，既妨碍主服务器又妨碍根服务器。BIND 使用一个残缺服务器“penalty box”来帮助处理因为残缺授权造成的负载，但是 Microsoft 的服务器没有实现它。

找到残缺授权的途径有两种：复查日志文件，或者使用一种叫做 **doc** 的工具，**doc** 是“domain obscenity control（域名模糊控制）”的缩写。我们将在下一节看到一些 **doc** 的例子，但让我们先复查一些日志项。

许多网点把日志通道 `lame-servers` 指向/**dev/null**，不要为其他人的残缺授权所烦恼。只要您自己的域很干净，它自己不是一个残缺授权的来源或者牺牲品就行了。

在大约一个星期所产生的一个 3.5MB 的日志文件（级别为 info）中，超过三分之一的日志项是残缺授权造成的。其中的 16%涉及向根服务器查询可能不存在的域。一个很固执的用户竟向根服务器查询 tokyotopless.net 数百次。唉！这里有一个残缺授权的例子：

www.TopSage.com

```
Jan 29 05:34:52 ipn.caida.org named[223]: Lame server on 'www.games.net' (in
    'GAMES.net'?): [207.82.198.150].53 'NS2.EXODUS.net'
```

下面看看我们是怎样用 **dig** 来跟踪这个问题的（我们删去了一些 **dig** 的过多输出）：

```
$ dig www.games.net.
;; ...
;; QUESTIONS:
;;      www.games.net, type = A, class = IN
;; ANSWERS:
www.games.net.      3600      A    209.1.23.92
;; AUTHORITY RECORDS:
games.net.          3600      NS   ns.exodus.net.
games.net.          3600      NS   ns2.exodus.net.
games.net.          3600      NS   ns.pcworld.com.
;; ADDITIONAL RECORDS: ...
```

本地服务器上的第一次查询返回了 www.games.net 的地址记录和权威服务器的列表。当我们查询 ns.exodus.net 处的服务器时，它工作正常（没有给出来），但 ns2.exodus.net 却是另一回事儿：

```
$ dig @ns2.exodus.net www.games.net.
;; QUESTIONS:
;;      www.games.net, type = A, class = IN
;; AUTHORITY RECORDS:
net.                244362    NS   F.GTLD-SERVERS.net.
net.                244362    NS   J.GTLD-SERVERS.net.
net.                244362    NS   K.GTLD-SERVERS.net.
net.                244362    NS   A.GTLD-SERVERS.net.
;; ...
```

ns2 被列为这个域的一台授权服务器，但是它却没有返回记录，从而把我们引导到顶级域 net 的服务器。因此，我们可以做出结论，ns2.exodus.net 配置不正确。

有时当您为了找到残缺授权的原因而 **dig** 一个权威服务器的时候，**dig** 什么信息都没有返回。再带**+norecurse** 标志查询一次，您就能看到这台服务器到底都知道什么。

15.16.8 doc：域名模糊控制

doc 不是 BIND 的一部分。它目前由 Brad Knowles 维护，从他的网站上可以下载 **doc**：

www.shub-internet.org/brad/dns （注意，是“shub”不是“shrub”。）

doc 是 C shell 脚本。如果您打算把它放到搜索路径中，或者从 **cron** 运行它，就必须编辑这个脚本，设置变量 auxd 指向安装路径。

doc 通过重复调用 **dig** 检查授权。它报告不一致性、错误和其他与一个特殊域名有关的问题。它的屏幕输出总结了它所发现的问题，它还会在当前目录下生成一个详细的日志文件。

让我们看几个例子。首先是 atrust.com，一切似乎都没问题：

```
$ doc atrust.com
Doc-2.2.3: doc atrust.com
Doc-2.2.3: Starting test of atrust.com.   parent is com.
Doc-2.2.3: Test date - Thu Jul  6 08:54:38 MDT 2006
Summary:
  No errors or warnings issued for atrust.com.
Done testing atrust.com.  Thu Jul  6 08:54:46 MDT 2006
```

日志文件（**log.atrust.com.**）提到，com 和 atrust.com 在 atrust.com 的名字服务器身份上保持一致，

www.TopSage.com

而且那两台服务器的 SOA 记录里序列号相同。**doc** 还检查每台名字服务器的 PTR 记录。由 **doc** 实际执行的 **dig** 查询和它们做出的反映都列在日志文件的末尾。

我们的下一个例子是 cs.colorado.edu 这个域的检查结果：

```
$ doc cs.colorado.edu
Doc-2.2.3: doc cs.colorado.edu
Doc-2.2.3: Starting test of cs.colorado.edu.   parent is colorado.edu.
Doc-2.2.3: Test date - Thu Jul  6 08:55:15 MDT 2006
dig: Couldn't find server 'rs0.netsol.com.': No address associated with hostname
DIGERR (NOT_ZONE): dig @rs0.netsol.com. for SOA of parent (colorado.edu.) failed
DIGERR (NOT_AUTHORIZED): dig @pacifier.com. for SOA of cs.colorado.edu. failed
dig: Couldn't find server 'xor.com.': No address associated with hostname
DIGERR (FORMAT_ERROR): dig @xor.com. for SOA of cs.colorado.edu. failed
Summary:
   ERRORS found for cs.colorado.edu. (count: 2)
   Incomplete test for cs.colorado.edu. (2)
Done testing cs.colorado.edu.  Thu Jul  6 08:55:26 MDT 2006
```

我们在这儿可以看到父域（colorado.edu）的几个问题。父域的区数据把 rs0.netsol.com 不正确地列为 colorado.edu 的一个服务器，而且把 xor.com 和 pacifier.com 不正确地列为 cs.colorado.edu 的服务器。要看到完整的信息，我们必须对父域（colorado.edu）执行 doc 命令，并且检查详细的日志文件。

最后，我们看一下 nsa.cctldmd.net：

```
$ doc nsa.cctldmd.net
Doc-2.2.3: doc nsa.cctldmd.net
Doc-2.2.3: Starting test of nsa.cctldmd.net.   parent is cctldmd.net.
Doc-2.2.3: Test date - Thu Jul  6 08:56:20 MDT 2006
SYSerr: No servers for nsa.cctldmd.net. returned SOAs ...
Summary:
   YIKES: doc aborted while testing nsa.cctldmd.net.  parent cctldmd.net.
   WARNINGS issued for nsa.cctldmd.net. (count: 2)
   Incomplete test for nsa.cctldmd.net. (1)
Done testing nsa.cctldmd.net.  Thu Jul  6 08:56:22 MDT 2006
```

cctldmd.net 这个域有一个 nsa.cctldmd.net 残缺授权。**log.nsa.cctldmd.net.**这个文件里的一条日志解释了这个问题：

```
WARNING: register.dns.md. claims to be authoritative for nsa.cctldmd.net.
 == but no NS record at parent zone
```

如果您管理的域包括子域（或者不信任父域的管理人），那么考虑每周从 **cron** 运行一次 **doc**，核实所有同您的域有关的授权都正确。

15.16.9 其他 DNS 检查工具

其他还有几种工具可以检查 DNS 环境的各个不同方面。BIND 9 自带了 **named-checkconf** 和 **named-checkzone**，它们做 **named.conf** 文件和区文件的基本语法（而不是语义）检查。原来 DNS 的检查工具是 Craig Leres 在劳伦斯伯克利实验室（Lawrence Berkeley Labs）写的 **nslint**。（从和 **doc** 一样的网站下载的）工具 **lamers** 会整合多个日志文件的内容，向违规网点的系统管理员发电子邮件，告诉他们有一个残缺授权，并说明如何解决这个问题。Jorge Frazao 和 Artur Romao 编写的 DDT 可以调试被缓存的数据。

www.TopSage.com

dnswalk 遍历授权树，找出父域和子域之间，或者正向和反向记录之间不一致的地方。它还能找到遗漏的点（.）、不必要的粘合记录等。它是一种保护 DNS 健康的通用工具。为了发挥 **dnswalk** 的神奇功效，它需要能做区传送。

15.16.10 性能问题

BIND 9 目前还有几个性能上的问题正在由 ISC 解决。它的性能接近了 BIND 8 的性能，但它在多处理机体系结构下的效率至今仍然令人失望。f.root-servers.net 这台机器使用 BIND 9，能轻松处理每秒 5 000 次查询，对大多数站点来说，这个性能可能还不错。

我们已经说过多次，但还要不厌其烦地重复一遍：把您的 TTS 设为合理的值（几个星期或者几天，不要几个小时或者几分钟）。使用的 TTL 太短都会降低您（因为您必须不停地重复提供同样的记录）和您 Web 客户机（因为它们必须不停地去取它们）的性能。

内存调页会非线性地降低服务器的性能，所以不要在运行 **named** 的服务器上舍不得加内存。要看到 **named** 内存用量稳定下来，您需要等上大约一周时间，参考 15.11.1 节的内容。

使用转发器。参考 15.11.4 节了解对转发体系结构的讨论。

15.17 发行版本的特定信息

本节介绍每种 Linux 发行版本所带的 **named** 软件及其默认配置的信息。当 ISC 发布 BIND 软件的新版本时，这些 Linux 发行版本都能很好地更新自己的 BIND 版本。

我们援引的 Linux 发行版本中的每一种都把随带的 BIND 作为默认的名字服务器软件，但只有一部分在默认安装里包括 BIND。其他发行版本带有 BIND 的工具程序（**dig**、**nslookup**、**host** 等），但却没有名字服务器软件本身。当您安装机器或者单独安装 BIND 软件包的时候，不得不指出自己想要安装 BIND。

作为 BIND 客户机的 Linux 机器在文件**/etc/resolv.conf** 中指定了它的默认域、在查询非完整域名的时候要搜索的其他域名，以及它的本地名字服务器的 IP 地址。充当 BIND 服务器的 Linux 主机也使用表 15.16 中列出的 **named** 文件。

表 15.16 Linux 中的 BIND 文件

文　件	目　录[a]	说　明
resolv.conf	**/etc**	解析器库的配置文件
named, lwres	**/usr/sbin**	名字服务器的守护进程
named.conf	**/etc** **/etc/bind**	**named** 的配置文件（RHEL、Fedora 和 SUSE） **named** 的配置文件（Debian、Ubuntu）
named.pid	**/var/run/named** **/var/run/bind/run** *home*/**var/run/named**	正在运行的 **named** 的进程 PID（RHEL/Fedora） 正在运行的 **named** 的进程 PID（Debian、Ubuntu） 正在运行的 **named** 的进程 PID（SUSE）
named.run	*home*	调试模式的输出
named.stats	*home* *home*/**var/log**	统计输出 统计输出（SUSE）
named_dump.db	*home* *home*/**var/log**	整个数据库的转储 整个数据库的转储（SUSE）

a. 我们用 *home* 指 **named.conf** 中指定作为 BIND 文件主目录的目录。

Linux 包含一个开关文件**/etc/nsswitch.conf**，它指定了应该怎样完成主机名到 IP 地址的映射，

www.TopSage.com

是否应该先（还是后，或者根本不）尝试DNS。如果没有这个开关文件，默认的做法是：

```
hosts: dns [!UNAVAIL=return] files
```

!UNAVAIL子句的意思是：如果有DNS，但是没有在那儿查到名字，那么查询尝试就应该算做失败，而不应该继续尝试下一项（在这里是/etc/hosts文件）。如果没有运行名字服务器（在启动的过程中可能会出现这种情况），那么查询进程就会参考**hosts**文件的内容。

我们举例的几个发行版本都提供了如下形式的默认**nsswitch.conf**文件：

```
hosts: files dns
```

对于配置查询来说，的确没有最好的方式——它取决于您的站点是怎样来管理的。一般而言，我们倾向于在DNS中，而不是在NIS或者普通文件中保存尽可能多的主机信息，但是我们仍然要保留必要时在系统引导期间能退回到使用静态**hosts**的做法。



安装RHEL或者Fedora的BIND软件包（目前是**bind-9.2.3-13.i386.rpm**）将二进制程序安装到**/usr/sbin**下，将手册页安装到**/usr/share/man**下，并且增加了一个叫做“named”的用户的组，而且为区文件创建了目录。“named”用户能够通过组权限访问到数据文件。

除非您在**/etc/sysconfig/named**做了修改，否则配置文件**named.conf**的位置在**/etc**下（就像Paul vixie和God希望的那样），区文件在**/var/named**下。虽然没有提供样板文件，但是**bindconf**软件包应该有这些文件。



Debian和Ubuntu能让您选择安装BIND 8还是BIND 9，这取决于您安装的软件包（**bind**或**bind9**）。下面的内容假定安装了BIND 9。

程序和文件的属主是root，而属组是“bind”，而且设定的权限即使是以“bind”用户而不是root的身份调用**named**执行，它也能访问到。

在**/etc/bind**里提供了一些有用的样板文件。其中包括**named.conf**的样本文件以及根索引、localhost、广播地址和私有地址空间的区文件。Debian提供的**named.conf**包含了**named.conf.options**和**named.conf.local**两个文件。它把BIND的默认目录定为**/var/cache/bind**，Debian里就有这个目录，但它是空的。

在**/etc**下的配置信息和**/var**下的区信息背后的逻辑是：如果您这里是其他网点的次级服务器，那么就不用控制**named**写的区文件大小，所以您可能想要把文件放在**/var**里。您作为主服务器的区也可以与这些配置文件共存，并且使用**named.conf**文件里的绝对路径，或者它们也可以在**/var/cache/bind**里。

如果您想要运行一台仅做缓存的服务器，那么不需要修改Debian的**named.conf**文件样本。如果您对某些区有授权，那么必须加上它们，最好把它们加入**named.conf.local**文件里。

Debian提供的样板文件使用了一些BIND新特性，有助于您成为网络上良好的DNS成员。例如，它们把com和net区配置为delegation-only区，避免让您的用户因为敲错什么而通过VeriSign的Site Find工具给它创造广告收益。如果您没有在内部使用RFC 1918的私有地址空间，那么可以清空RFC 1918区文件，防止这些地址从本地网络跑出去。Debian真不错！

目录**/usr/share/doc/bind9**包含几个有用的参考文献。找到**README.Debian**文件（甚至在Ubuntu上也是这么叫的），了解Debian在配置BIND上采取的策略。

www.TopSage.com



在 SUSE 上从一个 **rpm** 软件包安装 BIND 会让人感到沮丧，因为缺乏很多依赖包，但是解决掉这些问题之后，安装就变得很顺利了。安装过程会说明正在做什么，并且结果会装好一个能够使用且文档全面的名字服务器。一位 SUSE 向导后来建议使用 YaST 来代替 rpm 作为 SUSE 真正安装软件的方法，那或许是正确的答案。

在默认情况下，**named** 以“named”用户和组的身份，运行在用 **chroot** 把根目录改为 **/var/lib/named** 的环境下。安装过程建立了这个 **chroot** 的监管目录（jail）并且在里面放置了运行 **named** 需要的所有文件，甚至还有像 syslog 的 UNIX 域套接口这样的细节。额外的配置文件（不是 **named.conf**）和区文件在**/etc/named.d** 下，并且在 **named** 启动后复制到这个监管目录里。如果您不想让 **named** 在 **chroot** 后的环境里运行，那么可以修改**/etc/sysconfig/named** 里的这行：

```
NAMED_RUN_CHROOTED="yes"
```

您要改的就这么多。**/etc/init.d** 里的引导脚本要用到这个信息，而且能够以两种方式中的任何一种启动 **named**。

SUSE 提供了一个**/etc/named.conf** 样板文件，其中带有帮助性的注释，解释了许多选项的含义。SUSE 上**/etc/named.conf** 文件的权限一般不是所有人都能读。这个默认文件导入一个叫做 **named.conf.include** 的文件，接下来又从**/etc/named.d** 导入文件 **rndc-access.conf**，这两个文件所有人都能读。SUSE 在这里考虑到了什么样的安全问题还不完全清楚。**rndc** 预先配置为只接受从 localhost 来的控制命令。

SUSE 的 **named.conf** 文件可以原封不动地用来运行一个仅缓存的服务器。如果想要提供您自己区的域名服务，那么可以把区文件放在**/etc/named.d** 目录下，并且在 **/etc/named.conf.include** 文件里列出这些区的名字。

ISC 的 BIND 文件位于**/usr/share/doc/packages/bind9** 下。

15.18 推荐读物

介绍 DNS 和 BIND 的资料可谓多种多样，其中包括随发行版本带的文档、几本书中介绍 Internet 主题的章节、O’reilly Nutshell 系列中的一本书，以及各种在线资源。

15.18.1 邮件列表和新闻组

下面的邮件列表都和 BIND 有关：

- bind-announce——向地址 bind-announce-request@isc.org 发邮件就可以加入；
- namedroppers——向 namedroppers-request@internic.net 发邮件就可以加入；
- bind-users——向 bind-users-request@isc.org 发邮件就可以加入；
- bind9-workers——向 bind9-workers-request@isc.org 发邮件（供熟悉代码的人士使用）。

把 bug 报告发送到 bind9-bugs@isc.org。

15.18.2 书籍和其他文档

THE NOMINUM BIND DEVELOPMENT TEAM. *BINDv9 Administrator Reference Manual.*
从 www.isc.org 提供的 BIND 软件发布中能够得到这份文档（**doc/arm**）。

www.TopSage.com

这一文档概述了 BIND 9 的管理和运用。ISC 正在编写一个新版 BIND 文档，*BIND Reference Manual*（可能叫 **doc/brm**），但还看不到它。一种早期的文档 *Bind Operations Guide* 或者也称为 BOG，详细描述了 BIND 4 的操作和配置。BOG 一直都被包括在 BIND 8 的软件发布中。

ALBITZ, PAUL, AND CRICKET LIU. *DNS and BIND, (5th Edition).* Sebastopol, CA: O'Reilly, *2006.*

这本有关 BIND 的流行又受到推崇的书介绍了 BIND 8 和 BIND 9 的内容。它非常全面。

15.18.3 网上资源

DNS Resources Directory 的地址是 www.dns.net/dnsrd，它收集了有用的资源和资源链接，András Salamon 维护着它。

Google 在 directory.google.com/Top/Computers/Internet/Protocols/DNS 提供了一份 DNS 的资源索引。

15.18.4 RFC

定义 DNS 系统的 RFC 可以从 www.rfc-editor.org 获得。早期的和演变中的思路首先出现在 Internet-Drafts 系列中，以后移到 RFC 系列中。我们在本书上一版里列出了大约一页与 DNS 有关的最重要的 RFC，但是现在的 RFC 有那么多（超过 100 个，而且还有 50 个 Internet draft），所以您最好搜索 rfc-editor.org 这个网页，获得整个资料库。参考当前 BIND 软件发布的 doc/rfc 和 doc/draft 目录就看到所有的文档。

最初的、定义性的标准（1987 年发表）有：

- 1034—Domain Names: Concepts and Facilities（域名：概念和功能）；
- 1035—Domain Names: Implementation and Specification（域名：实现和规范）。

15.19 习题

E15.1 说明下面这些 DNS 记录每一种的功能：SOA、PTR、A、MX 和 CNAME。

E15.2 什么是粘合记录？为什么需要它们？使用 **dig** 找到把您的本地区和它的父区联系起来的粘合记录。

E15.3 否定缓存意味着什么？它为什么重要？

E15.4 为您的站点创建 SPF 伪记录，帮助控制垃圾邮件。

★E15.5 设立一个新的二级域需要哪些步骤？包括技术和程序（译者注：指办事程序）两方面的因素。

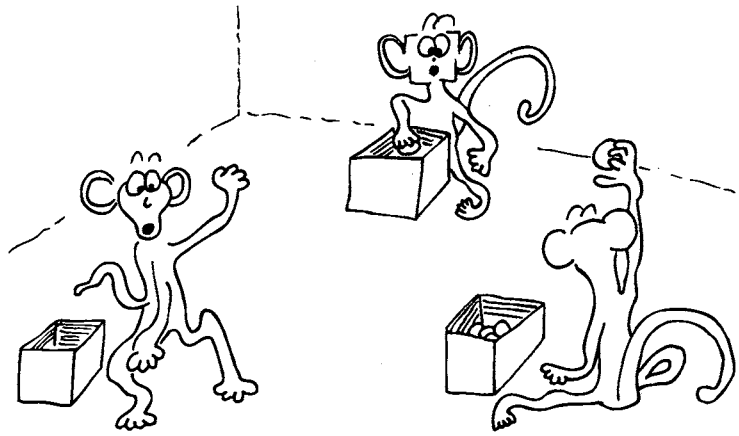
★E15.6 对一次 DNS 查询的权威和非权威回答之间有什么不同？您怎样确保回答是权威的？

★E15.7 您本地的名字服务器是什么机器？解析 www.admin.com 这个名字必须有哪些步骤？假定在 DNS 的任何地方都没有有关该域的缓存信息。

★E15.8 说明 DNS 的 UDP 包长有 512 字节限制的意义。解决这个潜在问题可以使用什么样的变通方法？

www.TopSage.com

第 16 章　网络文件系统



网络文件系统（Network File System）通常叫做 NFS，它可以在计算机间共享文件系统。NFS 对用户几乎是透明的，并且是“无状态的”，这意味着当 NFS 服务器崩溃时不会丢失任何信息。客户机只是等着服务器恢复正常，然后就好像什么也没有发生过一样继续工作。

NFS 由 Sun Microsystems 公司在 1985 年推出。实现它原本是作为无盘客户机的一种替代文件系统，但这种协议经证实设计得很好，作为一种通用的文件共享解决方案非常有效。实际上，我们已经很难记得在 NFS 出现以前的情形是怎样的了。所有功能完善的 Linux 发行版本都支持 NFS。

16.1　NFS 概述

现如今，NFS 仅用于在 Linux 和 UNX 主机之间共享文件。Windows 客户机应该使用 CIFS/Samba 获得文件服务。参见 26.6 节了解有关 Samba 和 CIFS 的更多资料。

NFS 由许多部分组成，包括一种安装协议与安装服务器，协调基本文件服务的守护进程，以及几个诊断工具。服务器端和客户端软件都有一部分驻留在内核中。不过 NFS 的这些部分不需要配置，而且从管理员的角度来看，它们在很大程度上都是透明的。

16.1.1　NFS 协议的版本

NFS 协议一直以来都格外稳定。NFS 最初公开发行的版本是第 2 版。20 世纪 90 年代早期，这个协议融入了一些修改，产生了第 3 版，这个版本提高了性能，并对大文件提供了更好的支持。现在有了第 4 版的一些早期实现。第 4 版包括许多新的增强功能，我们将在下面予以介绍。

www.TopSage.com

因为 NFS 第 2 版的客户机只有在接收到服务器的确认信息之后才能确认一次写操作已经完成，所以第 2 版的服务器在作出应答前，必须向磁盘提交每个被修改的块，以避免在发生崩溃时出现矛盾。这种限制使得 NFS 的写操作有很大的延迟，因为在正常情况下，被修改的块只需写入内存中的高速缓冲区。

NFS 第 3 版采用了一种保持一致性的机制，允许异步地执行写操作，从而消除了这个“瓶颈”。这个版本还更新了协议的其他几个方面，人们发现这些地方会造成性能上的问题。这样一来，NFS 第 3 版比第 2 版的运行速度要更快一点儿。NFS 第 3 版的软件一定能与第 2 版实现互操作，此时第 3 版只是返回去使用原来的协议而已。

NFS 第 4 版正在变得越来越稳定，有些 Linux 的版本提供了这个版本的 NFS。它要求 2.6.1 版以后的 Linux 内核，而且需要手工开启内核中的支持。

NFS 第 4 版的增强功能有：

- 能与防火墙和 NAT 设备兼容且互操作；
- 核心 NFS 协议内集成了上锁（lock）和安装（mount）协议；
- 有状态操作；
- 集成有很强的安全措施；
- 支持复制和迁移；
- 支持 UNIX 和 Windows 客户机；
- 支持访问控制列表（ACL）；
- 支持 Unicode 文件名；
- 在低带宽连接下具有良好性能。

参见 nfs.sourceforge.net 了解有关 Linux 上 NFSv4 当前进展状态的信息，还有 NFS 最新软件发布的链接。

因为 NFSv4 仍处于开发之中，我们在本章没有详细讨论它。但是您可以保持关注，许多计划实现的功能解决了 NFS 长期存在的一些缺点。我们希望第 4 版能够实现它所提出的那些好功能。

16.1.2 传输协议的选择

NFS 运行在 Sun 的 RPC（Remote Procedure Call，远程过程调用）协议之上，该协议定义一种与系统无关的方法来实现进程的网络通信。这种体系结构的一个优点就是既可以用 UDP 也可以用 TCP 作为下层的传输协议。

NFS 起初使用 UDP，因为 UDP 在 20 世纪 80 年代的 LAN 和计算机上性能最好。虽然 NFS 确实会对自己的包序列进行重组与错误检查，但是 UDP 和 NFS 都缺少拥塞控制算法，而这对于在一个大型 IP 网络上获得高性能来说是必不可少的。

为了解决这些潜在的问题，所有的现代操作系统现在都能使用 TCP 而不是 UDP 来作为 NFS 的传输协议。这种选择首先是作为帮助 NFS 通过路由器在 Internet 上工作的一种手段得以应用的。然而，目前的调查发现，TCP 也是本地 NFS 通信的最好选择。随着时间的推移，原来选择 UDP 而不是 TCP 的大多数理由都因为快速 CPU 和廉价内存以及更加智能化的网络控制器的出现而不再成为理由。Linux 从 2.4 内核开始就支持用 TCP 的 NFS 服务。

大多数支持 TCP 的服务器一般能接受两种传输协议的连接，所以选择 TCP 还是 UDP 是由客户机来决定。客户机通过 **mount** 命令的选项（手工安装 NFS 文件系统的情况下），或者在**/etc/fstab** 这样的配置文件里指出自己的选择。

16.1.3 文件上锁机制

文件上锁（由系统调用 **flock**、**lockf** 以及 **fnctl** 实现）在很长时间内一直是 UNIX 系统令人头疼的

www.TopSage.com

问题。在本地文件系统上，人们已经知道它的运行不太理想。在 NFS 环境下更是不可靠。根据设计，NFS 服务器是无状态的：即它们不知道哪些机器正在使用某个特定文件。然而，实现文件上锁需要这样的信息。怎么办呢？

传统的解决方法是独立于 NFS 实现文件上锁。大多数操作系统提供两种守护进程（即 **lockd** 和 **statd**），以便能把文件上锁功能搞好一些。遗憾的是，这一任务由于多种敏感的原因做起来很困难，所以 NFS 文件上锁一直是个棘手问题。

16.1.4 磁盘配额

要获得远程磁盘配额信息，可以用一种带外服务程序 **rquotad**。如果下层的文件系统启动了磁盘配额功能，那么 NFS 服务器也能实施磁盘容量的配额，但用户只有在远程服务器运行 **rquotad** 时才能查看到他们得到的配额情况。

我们认为磁盘配额的做法基本上过时了，但是有些单位仍然要靠它们防止用户占满磁盘空间。如果您在这样的单位里工作，那么可以参考针对 Linux 发行版本的磁盘配额文档。我们不再进一步讨论 **rquotad** 的用法。

16.1.5 cookies 与无状态安装

客户机在使用 NFS 前必须显式地安装 NFS 文件系统，就像客户机必须安装保存在本地磁盘上的文件系统一样。然而，因为 NFS 是无状态的，所以服务器不知道各个文件系统被哪些客户机已经安装到本地了。服务器只是在同客户机进行了一次成功的安装协商之后，透露给客户机一个秘密的 cookie。这个 cookie 标识出 NFS 服务器上已经被安装的目录，这样就为客户机提供了一条能访问其内容的途径。

卸装与重新安装服务器上的文件系统时通常要改变它的 cookie。这里有一个特殊的情况，即系统重新启动后 cookie 保持不变，于是崩溃的服务器能重返它先前的状态。但是，不要在启动系统后进入单用户模式，用完文件系统后又再次重启系统。这个过程将撤销 cookie，使客户机在重新启动或重新安装文件系统之前，都不能访问它们已经安装过的文件系统。

一旦客户机有了一个神奇的 cookie，它就可以利用 RPC 协议请求执行对文件系统的操作，如创建一个文件或读取一个数据块。因为 NFS 是一种无状态协议，所以要由客户机来负责确保在它删除要写入服务器的数据副本之前，服务器已经确认了写请求。

16.1.6 共享文件系统的命名规则

如果在安装文件系统的时候采用一种命名方案，让名字里包括每个远程服务器的名字（例如，对于服务器 **anchor** 上的某个文件系统，就用**/anchor/tools**），那么管理 NFS 就会更容易。照这样起名字很有用，因为它们能让用户将诸如“**anchor** 会在周六因升级而整天停机”这样的通知理解为“我在周六不能使用**/anchor/tools/TeX** 完成我的论文，所以我应该出去滑雪”。

遗憾的是，这种方案需要在所有客户机的根目录下都存在**/anchor** 目录。如果一台客户机从别的几台主机上得到文件系统，则客户机的根目录下会很混乱。可以考虑提供一种更深层次的结构（如**/home/anchor**，**/home/rastadon** 等）。我们建议使用 16.6 节介绍的自动安装守护进程之一来实现这样一种方案。

16.1.7 安全性与 NFS

NFS 提供了一种访问网络上文件的便利途径，因此它引起安全问题的可能性很大。NFS 第 3 版在

www.TopSage.com

许多方面都像是一个捣乱的孩子，给 UNIX 和 Linux 安全带来很多问题。NFS 协议最初设计时没有特别重视安全问题——方便是要付出代价的。幸好 Linux 支持许多特性，能够减少和隔离 NFS 的传统安全问题。

对 NFS 卷的访问是由一个名为**/etc/exports** 的文件来批准的，这个文件枚举了若干系统的主机名（或者 IP 地址），这些系统应该有权访问服务器上的文件系统。遗憾的是，这在安全上是一种很薄弱的形式，因为服务器相信客户机告诉它自己是谁。客户机很容易向服务器撒谎，所以这种机制不是完全可信的。然而，您应该只向您信任的客户机导出文件系统，而且您应该一直检查，保证没有把文件系统向整个世界导出。参考 16.2.1 节了解有关文件导出的更多知识。

要严格限制对 NFS 导出文件系统的访问。幸运的是，所有的 Linux 版本都包括一个能够完成这个任务的防火墙。

和对本地文件系统一样，对 NFS 文件系统的文件级访问控制是按照 UID、GID 和文件权限来管理的。但是 NFS 服务器依然会信任客户机告诉它的，谁正在访问文件的消息。如果 mary 和 bob 在两台不同的客户机上都有同样的 UID，那他们就可以访问到彼此的 NFS 文件。此外，在系统上有 root 权限的用户还可以把 UID 改成他们想要的任何值，服务器都会乐于赋予它们对相应文件的访问权限。出于这些原因，我们强烈建议到处都使用唯一的 UID，而且使用下一节所介绍的 root_squash 选项。

我们知道有一家相当大的教育单位因为失误没有使用 root_squash 选项。结果，5 台大服务器和 60 台桌面台式机被入侵。要花一个长长的假日周末来控制安全事件，重建这些机器。

如果您的网点已经安装了一个网络防火墙，禁止访问 NFS 使用的 TCP 和 UDP 2049 端口是个不错的想法。您还应该禁止对 **portmap** 守护进程的访问，该程序通常监听 TCP 和 UDP 端口 111。NFS 文件系统不应该导出给非本地的机器，或者通过开放的 Internet 对外导出文件系统，这点应该暗含在安全预防措施中，但或许还是值得明确地说出来。

16.1.8 root 访问与 nobody 账号

即使应该赋予用户对各处相同的访问权限，但传统上都会避免 root 在通过 NFS 安装的文件系统上随便运行。默认情况下，Linux 的 NFS 服务器截获以 UID 0 名义传来的请求，将其改变成好像是来自某些别的用户。这种修改称为“squashing root（压制 root）”。因此 root 账号不是完全关闭的，但会限制 root 只拥有普通用户的访问权限。

特意定义了一个占位账号 nobody，它是一个远程的 root 在 NFS 服务器上假装的“用户”。传统上给 nobody 的 UID 是 65 534（-2 的补码）[1]。您可以用导出选项 `anonuid` 和 `anongid` 改变 root 默认的 UID 和 GID 映射。您可以使用 `all_squash` 选项将所有客户的 UID 映射到服务器的同一个 UID。这样的配置消除了用户之间的一切区别，带来了一种对文件系统的公共访问。

在另一个极端，`no_root_squash` 选项关闭了 root 的 UID 映射。有时需要这个选项来支持无盘客户机或者要求对文件有 root 访问权的软件。一般而言，关闭这项功能不是个好主意，因为它能让在客户机上有 root 权限的用户修改正常情况下受到保护的文件。不过，确实是有这么一个选项。

这些预防措施背后的意图是好的，但它们最终达到的效果并没有设想的好。NFS 客户机上 root 账号能利用命令 **su** 变成任何它想变成的 UID，因此用户文件其实没有受到保护。系统账号（如 bin 和 sys）没有做 UID 映射，所以属于它们的文件（占系统文件的很大比例）很容易受到攻击。UID 映射的唯一实际作用只是避免访问 root 拥有的文件以及不能为外界读或写的文件。

[1] 虽然 Red Hat 的 NFS 服务器默认取 UID -2，但是 **passwd** 文件中的 nobody 账号使用的 UID 却是 99。您可以不去管它，在 **passwd** 文件加入一个 UID 是-2 的项，或者，如果您愿意，可以把 `anonuid` 和 `anongid` 改为 99。其他发行版本在 **passwd** 文件中用于 nobody 和 nogroup 账号的 UID 照样是-2。

www.TopSage.com

16.2 服务器端 NFS

当服务器让某个目录能为其他计算机所使用时，通常说这个服务器“导出（export）”了一个目录。

在 NFS 第 3 版中，客户机安装文件系统（也就是说，获知它的秘密 cookie）所用的过程完全不同于访问文件所用的过程。这两种操作使用不同协议，请求由不同的守护进程来提供服务：**mountd** 服务于安装请求，而 **nfsd** 服务于实际的文件服务。有些系统把这两个守护进程称为 **rpc.nfsd** 和 **rpc.mountd**，表明系统所依赖的下层协议是 SunRPC（从而要求运行 **portmap**，参考 29.3 节）。为了阅读方便起见，我们略去了后缀 **rpc**。

在 NFS 服务器上，**moutd** 和 **nfsd** 都应该在启动系统的同时启动，而且只要系统开着就得让它们保持运行。如果您已经做了“导出（exports）”配置，那么系统启动脚本一般会自动启动这些程序。表 16.1 给出了每种平台上 NFS 服务器启动脚本的名字。

表 16.1 NFS 服务器的启动脚本

发 行 版 本	启动脚本的路径
Red Hat Enterprise	**/etc/rc.d/init.d/nfs**
Fedora	**/etc/rc.d/init.d/nfs**
SUSE	**/etc/init.d/nfsboot**[a]
Debian	**/etc/init.d/nfs-kernel-server** **/etc/init.d/nfs-common**
Ubuntu	**/etc/init.d/nfs-kernel-server** **/etc/init.d/nfs-common**

a. 在 SUSE 上，**/etc/init.d/nfs** 安装 NFS 客户文件系统。

mountd 与 **nfsd** 共享一个访问控制数据库，该数据库说明了哪些文件系统应该被导出，而哪些客户机有可能会安装它们。除了内核内部的表之外，这个数据库的运行副本通常还保存在一个称为 **/usr/lib/nfs/xtab** 的文件中。因为 **xtab** 并不是供普通人阅读的，所以您要用一条辅助命令来添加和修改其中的项：**exportfs**。要从导出表里删除项，可以使用 **exportfs -u**。

在大多数系统上，**/etc/exports** 是导出目录的说明清单，普通人可以直接阅读。在默认情况下，**/etc/exports** 中的所有文件系统都可以在系统引导时被导出。您可以使用 **exportfs -a** 命令手工导出 **/etc/exports** 中列出的全部文件系统，这应该在您对 **exports** 文件进行过修改之后运行。您还可以在每次导出文件系统的时候直接在 **exportfs** 的命令行上指出客户机、路径和选项。

NFS 是在逻辑层面上处理文件系统的。任何目录均可导出，导出目录不必是一个安装点或一个物理文件系统的根。然而，为了安全起见，NFS 特别注意文件系统之间的界限，并要求每个设备独立地导出。例如，在一台有 **/users** 分区的机器上，可以导出根目录但不会导出 **/users**。

通常，如果客户机愿意，它也可以安装导出目录下的子目录，但协议并不要求有这个特性。例如，如果一台服务器导出了 **/chimchim/users**，某个客户机可以只安装 **/chimchim/users/joe**，并且忽略 **users** 目录下的其他内容。大多数 UNIX 的版本不允许以不同选项导出一个已导出的目录下的子目录，但 Linux 却可以。

16.2.1 exports 文件

/etc/exports 文件列举出通过 NFS 导出的文件系统，以及可以访问每个文件系统的客户机。客户机列表和文件系统之间以空白分隔，每个客户机之后紧跟着用括号括起来、以逗号分隔的一系列选项。

www.TopSage.com

用反斜线可以续行。

下面是这个文件格式的样子：

```
/home/boggs      inura(rw,no_root_squash)  lappie(rw)
/usr/share/man   *.toadranch.com(ro)
```

虽然有些客户机的说明实际上指的是多台主机，但没有办法为一组选项列出多台客户机。表 16.2 列出了 **exports** 文件里可以出现的 4 种类型的客户机说明。

表 16.2 /etc/exports 文件中客户机的说明

类 型	语 法	含 义
主机名	*hostname*	单个主机
网络组	*@groupname*	NIS 网络组，参考 17.4.4 节了解细节
通配符	*和?	具有通配符的 FQDN[a]，“*”不匹配点号（.）
IP 网络	*ipaddr/mask*	CIDR 风格的说明（如 128.138.92.128/25）

a. 代表完整域名（Fully qualified domain name）。

表 16.3 给出了最常用的导出选项。

表 16.3 常用的导出选项

选 项	说 明
ro	以只读方式导出
rw	以读写方式导出（默认方式）
rw=*list*	大多数客户机为只读。*list* 举出的主机允许以可写方式安装 NFS，其他所有主机必须以只读方式安装
root_squash	将 UID 0 和 GID 0 映射（“压制”）成 anonuid 和 anongid 所指定的值
no_root_squash	运行 root 正常访问，这很危险
all_squash	将所有的 UID 和 GID 映射到它们各自的匿名版本上。用于支持 PC 和不可信的单用户主机
anonuid=*xxx*	指定远程 root 账号应被映射的 UID 号
anongid=*xxx*	指定远程 root 账号应被映射的 GID 号
secure	远程访问必须从授权端口发起
insecure	允许从任何端口远程访问
noaccess	防止访问这个目录及其子目录（用于嵌套导出）
wdelay	为合并多次更新而延迟写入磁盘
no_wdelay	尽可能快地把数据写入磁盘
async	让服务器在实际写磁盘之前就先回应写请求
nohide	暴露在导出的文件树中安装的文件系统
hide	与 nohide 的作用相反
subtree_check	验证每个被请求的文件都位于导出的目录子树里
no_subtree_check	只验证涉及被导出的文件系统的文件请求
secure_locks	要求所有的上锁请求都要有授权
insecure_locks	指定较为宽松的上锁条件（支持较老的客户机）
auth_nlm	与 secure_locks 同义
no_auth_nlm	与 insecure_locks 同义

Linux 的 NFS 服务器有个不寻常的特性，它允许已被导出的目录的子目录以不同的选项再被导出。它还提供了 noaccess 选项，不导出您不想共享的子目录。

www.TopSage.com

例如，配置：

```
/home          *.toadranch.com(rw)
/home/boggs    (noaccess)
```

能够让 toadranch.com 域内的主机可以访问**/home** 下除**/home/boggs** 之外的所有内容。在第二行没有客户机的名字出现，说明这个选项适用于所有主机，这样或许会更安全一些。

`subtree_check` 这个选项（默认选项）要验证客户机所访问的每个文件是否在一个被导出的子目录里。如果关闭了这个选项，那么只验证文件是否在一个被导出的文件系统之中。当被请求的文件在客户机打开它的时候被改了名字，那么子树检查功能偶尔也会带来问题。如果您预计到会出现许多这样的情况，那么就要考虑设置 `no_subtree_check`。

`secure_locks` 选项要求获得授权并经过身份验证才能对文件上锁。有些 NFS 客户机不会随上锁请求一起发送凭据，遇到 `secure_locks` 选项就无法使用 NFS 了。在这种情况下，只能对所有人都能读的文件上锁。最好的解决办法是把这些客户机替换为能正确支持凭据的客户机。不过，您可以指定 `insecure_locks` 选项来临时对付一下。

不要忘记在更新了 **exports** 文件之后运行 **exportfs -a** 让您所做的改动起作用。

16.2.2 nfsd：提供文件服务

一旦 **mountd** 证实客户机的安装请求是合法的，那么客户机就可以请求各种文件系统操作了。这些请求在服务器端由 **nfsd** 处理，**nfsd** 是 NFS 的运行守护进程[2]。除非客户机要导出自身的文件系统，否则 NFS 的客户机不必运行 **nfsd**。

nfsd 需要一个数值参数来指定它应该派生出多少个服务器线程。选择适当数目的 **nfsd** 很重要，遗憾的是，这是一件没有准头的事儿。如果数目太低或太高，NFS 的性能就会受到影响。

一般而言，对一台使用不频繁的服务器来说，有 8 个 **nfsd** 线程就够了，而且数目少的话，就不会出现性能问题。在一台生产服务器上，12~20 个 **nfsd** 是比较好的数目。随着 **nfsd** 数目的增加，您将发现平均负载会上升（正如 **uptime** 所报告的那样），说明 **nfsd** 的数目太多，应该从这个域值处稍微减少一点。您还应该定期地运行一下 **nfsstat**，检查 **nfsd** 线程的数量可能引发的性能问题。参考 16.4 节了解有关 **nfsstat** 的更多信息。

在带有许多 UDP 客户机的一台 NFS 服务器上，当所有 **nfsd** 线程都在使用，而此时又有请求到达的时候，就会发生 UDP 套接口溢出。您可以使用 **netstat -s** 命令监测溢出数目。增加更多的 **nfsd** 直到 UDP 套接口溢出数目降为 0。溢出表明服务器提供的守护进程数量不足，所以增加的 **nfsd** 要比通过这种方法测量出来的数目多几个。

要改变 **nfsd** 进程的数量，您可以编辑**/etc/init.d** 下恰当的启动脚本，或者在手工启动 **nfsd** 的时候在命令行指定。参考表 16.1，了解要编辑的脚本的名字。

16.3 客户端 NFS

安装 NFS 文件系统和安装本地磁盘上的文件系统非常相似。**mount** 命令能够理解 *hostname:directory* 这样的记法代表主机 *hostname* 上的目录 *directory*。和本地文件系统一样，**mount** 会把远程主机 *hostname* 上的目录 *directory* 映射到本地文件树上的一个目录。在完成安装之后，就可以像本地文件系统一样来访问

[2] 事实上，**nfsd** 只是对嵌在内核中的 NFS 服务器代码作一次没有返回的系统调用。

www.TopSage.com

以 NFS 方式安装的文件系统。**mount** 命令及其与 NFS 有关的扩展是 NFS 客户机上的系统管理员最关心的问题。

在一个 NFS 文件系统能够被安装之前，它必须正确导出（参见 16.2 节）。为了从客户机的角度来核实一台服务器已经正确地导出了它的文件系统，您可以使用客户机的 **showmount** 命令：

```
$ showmount -e coyote
Export list for coyote:
/home/boggs inura.toadranch.com
```

这个例子指出服务器 coyote 上的目录**/home/boggs** 已经被导出给客户系统 inura.toadranch.com。如果执行一次 NFS mount 操作没有起作用，而您已经用 **exportfs** 命令在服务器上核实过，文件系统已经正确地导出了，那么您应该首先检查 **showmount** 命令的输出（您可能会忘记在更新了 **exports** 文件之后运行一下 **exportfs -a**）。

如果服务器上已经正确地导出了目录，但是 **showmount** 返回了一条出错消息，或者返回的列表为空，您可能需要复查一下服务器上所有必须的进程是否都在运行（**portmap**、**mountd**、**nfsd**、**statd** 和 **lockd**），**hosts.allow** 和 **hosts.deny** 文件是否允许访问那些守护进程，以及您是否处在正确的客户机系统上。

要实际安装这个文件系统，您应该使用类似这样的一条命令：

```
# mount -o rw,hard,intr,bg coyote:/home/boggs /coyote/home/boggs
```

在**-o** 之后的选项指出了文件系统应该以读写方式来安装，操作应该是可以中断的，而且重试应该在后台完成。这些标志相当标准，表 16.4 列出了其他的常用标志。

表 16.4 NFS 的安装标志

标 志	说 明
rw	以读写方式安装文件系统（也必须要以相同的方式来导出）
ro	以只读方式安装文件系统
bg	如果安装失败（服务器没有响应），在后台一直尝试，继续发其他的安装请求
hard	如果服务器客机，让试图访问它的操作被阻塞，直到服务器恢复为止
soft	如果服务器客机，让试图访问它的操作失败，返回一条出错消息。这项功能对于避免进程“挂”在无关紧要的安装操作上来说非常有用
intr	允许用户中断被阻塞的操作（并且让它们返回一条出错消息）
nointr	不允许用户中断
retrans=*n*	指定在以软方式安装的文件系统上，在返回一条出错消息之前重复发出请求的次数
timeo=*n*	设置请求的超时时间（以十分之一秒为单位）
rsize=*n*	设置读缓冲的大小为 n 字节
wsize=*n*	设置写缓冲的大小为 n 字节
nfsvers=*n*	设置 NFS 协议为第 2 版或者第 3 版（在正常情况下是自动的）
tcp	选择通过 TCP 来传输。默认选择 UDP
async	让服务器在实际写磁盘之前就先回应写请求

以 hard（硬）方式安装的文件系统（这是默认情况）在它们的服务器宕机时会导致进程挂起。当牵扯到的进程是标准守护进程的时候，这种事情特别烦人，所以我们不建议通过 NFS 来提供对关键的系统二进制文件的服务。一般而言，这些选项都有它们各自不好的副作用，比如在一个 20 小时的仿真

www.TopSage.com

程序在已经运行了 18 个小时的时候中断了，仅仅是因为一次暂时性的网络波动[3]。自动安装方案，比如本章后面将要讨论到的 **autofs** 和 **amd**，也能提供针对这类安装毛病的一些补救办法。

读和写缓冲的大小对于 UDP 和 TCP 方式的 NFS 安装都适用，但是最优值是不一样的。因为您可以信任 TCP 来高效地传输数据，所以这个值应该高一些，取 32K 这个值比较好。对于 UDP 来说，在服务器和客户机都在同一个子网上的时候，8K 是个比较好的值[4]。虽然手册页建议把它增加到 8K 来获得更好的性能，但默认值却是 1K。

不过，在老的 Linux 2.2 和 2.4 内核里，默认的输入队列大小是 64K。当 NFS 服务器上运行有 8 个 **nfsd** 线程的时候，在 NFS 请求开始被丢弃之前，每个 **nfsd** 的实例只有一个没有完成的请求。因此，您可以考虑只增加 **nfsd** 的接收队列大小，在运行完 **nfsd** 之后再改回默认值，从而不会因为改动而给其他进程带来负面影响。您可以使用 **procfs** 在系统的启动脚本中改变输入队列的大小。

下面这个例子将队列设为 256K，它是一个合理的默认值。

```
rmem_default='cat /proc/sys/net/core/rmem_default'
rmem_max='cat /proc/sys/net/core/rmem_max'
echo 262144 > /proc/sys/net/core/rmem_default
echo 262144 > /proc/sys/net/core/rmem_max
```

运行或者重新启动 **rpc.nfsd**，然后把设定恢复到它们原来的值：

```
echo $rmem_default > /proc/sys/net/core/rmem_default
echo $rmem_max  > /proc/sys/net/core/rmem_max
```

用 **df** 可以测试 NFS 文件系统的安装情况，就像您测试一个本地文件系统一样：

```
$ df /coyote/home/boggs
Filesystem          1k-blocks   Used  Available  Use%   Mounted on
coyote:/home/boggs 17212156 1694128   14643692   11%   /coyote/home/boggs
```

NFS 分区可以用 **umount** 命令来卸装。如果在卸装 NFS 文件系统的时候，它正好在用，则会得到这样一条出错消息：

```
umount: /coyote/home/boggs: device is busy
```

和其他任何文件系统一样，一个 NFS 文件系统不能在它还在用的时候卸装。使用 **lsof** 命令找出在文件系统上打开了文件的进程，杀死它们，或者如果是因为 shell 的话，可以改变目录。如果别的办法都不行，或者您的服务器宕机了，尝试 **umount -f** 强行卸装该文件系统。

16.3.1 在启动时安装远程文件系统

您可以用 **mount** 命令建立暂时的网络安装，但应该在**/etc/fstab** 文件中列出是系统永久配置一部分的安装，以便在启动时可以自动地安装它们。另外，安装可由一种自动安装服务（如 **autofs**）处理，参考 16.6 节。

下面的 **fstab** 配置项从主机 coyote 和 inura 安装文件系统**/home** 和**/usr/local**：

[3] 我们的技术审稿人之一 Jeff Forys 说："大多数 NFS 安装都应该使用 hard、intr 和 bg 方式，因为这些选项最能保留 NFS 最初的设计目标（可靠性和稳定性）。soft 方式是一种令人讨厌的方式，是一种丑恶的扩展！如果用户想要中断，那自然没有问题。否则，也可以等着服务器恢复，一切都最终恢复原状，而且不会丢失数据。"

[4] 如果您使用了 **ipchains** 或者 **iptables**，您可能不得不加入一条规则来接收分片，因为 8K 超出了以太网的 MTU。接收分片可能会让您更容易受到拒绝服务（DoS）攻击。

www.TopSage.com

```
# filesystem      mountpoint    fstype  flags                          dump  fsck
coyote:/home      /coyote/home  nfs     rw,bg,intr,hard,nodev,nosuid    0     0
inura:/usr/local  /usr/local    nfs     ro,bg,intr,soft,nodev,nosuid    0     0
```

当您给 **fstab** 加上配置项的时候，要确保用 **mkdir** 创建适当的目录作为安装点。您可以通过运行 **mount -a -t nfs**，安装 fstab 中所有的 `nfs` 类文件系统，让改动立即起作用（不需要重新启动）。参考 7.4.6 节了解有关 **fstab** 文件的更多信息。

/etc/fstab 的 flags（标志）域指定了 NFS 安装的选项，这些选项和您要在 **mount** 命令行中指定的选项是一样的。

16.3.2 端口安全限制

NFS 客户机可以在连接一个 NFS 服务器时使用它们喜欢的任何 TCP 或 UDP 端口。然而，如果文件系统是用选项 secure 来导出的，那么 Linux 服务器就会坚持要求从客户机来的安装请求必须来自某个特权端口（端口号小于 1024）。对于 PC 和桌面 Linux 机器而言，使用特权端口实际上并不会增加多少安全性。

Linux 的 NFS 客户机采用传统的方法（仍然推荐该法），即从默认设置的特权端口访问服务器，这样可以避免潜在的冲突。要接受从非特权源端口来的安装，要用导出选项 insecure 导出文件系统。

16.4 nfsstat：转储 NFS 统计信息

nfsstat 命令能显示 NFS 系统保留的各种统计信息。**nfsstat -s** 显示 NFS 服务器进程的统计信息，**nfsstat -c** 显示与客户端操作相关的信息。例如：

```
$ nfsstat -s
Server rpc stats:
calls      badcalls   badauth    badclnt    xdrcall
24314112   311        9          302        0
Server nfs v2:
getattr    null       setattr    root       lookup     readlink
8470054 34% 58   0%   55199  0%  0      0%  1182897 4% 917    0%
read       wrcache    link       create     remove     rename
6602409 27% 0    0%   7452   0%  61544  0%  46712   0% 11537  0%
write      symlink    mkdir      rmdir      readdir    fsstat
7785789 32% 744  0%   3446   0%  2539   0%  13614   0% 69201  0%
```

这个例子取自于一个相当健康的 NFS 服务器。如果超过 3%的调用超时，那么 NFS 服务器或网络很可能出现了问题。检查 **netstat -s** 的输出了解总体的网络统计信息。它可能会暴露出丢包、分片重组，或者网络队列超限等影响 NFS 性能的问题。参考 19.4 节了解有关用 **netstat** 调试网络的更多信息。

偶尔运行一下 **nfsstat** 和 **netstat** 命令，熟悉一下它们的输出，有助于帮助系统管理员比普通用户早发现 NFS 出现的问题。

16.5 专用 NFS 文件服务器

快速而可靠的文件服务是任何生产计算环境（production computing environment）中最重要的环节

之一。虽然您肯定可以把一台 Linux 工作站和若干现成的硬盘组装成文件系统服务器，但这通常不是最有效和最容易管理的解决方案（即使它常常是最便宜的方案）。

专用 NFS 文件服务器已经有十几年的历史。与自制服务器的方法相比，它们提供的主机具有许多潜在的优势：

- 它们针对文件服务进行了优化，往往能提供尽可能好的 NFS 性能；
- 随着存储要求的增加，它们能平滑地扩展支持 TB 的存储量和数百个用户；
- 它们比 Linux 主机更可靠，因为它们有简化的软件、冗余的硬件、并且利用了硬盘的镜像技术；
- 它们往往既能为 Linux 客户机也能为 Windows 客户机提供文件服务，有些服务器甚至包含了集成的 Web 和 FTP 服务器；
- 它们与 Linux 文件服务器更容易管理；
- 它们往往提供了比普通 Linux 系统更优越的备份和检查工具。

我们最喜欢的一些专用 NFS 服务器包括 Network Appliance 公司的产品。该公司的服务器从小容量到大容量都有，而且价格合理。EMC 公司是高端服务器市场中的另一家厂商。其产品质量不错，但您要准备好接待不屈不挠的推销员，并为接受喋喋不休的市场宣传培养耐心[5]。

现在 SAN（storage area network，存储区域网络）服务器也非常流行。它们与专用 NFS 文件服务器的区别在于它们不知道文件系统的存在，它们只是提供磁盘存储块。因此，SAN 不会有操作系统上的开销，能够提供非常快的读写访问速度。此外，我们在实践中发现它们还不适用于生产环境。要把它们集成到已有的环境中相当复杂，需要获得很长时间的支持才行。

16.6 自动安装

在文件**/etc/fstab** 中列出各个文件系统，一次安装一个的做法，在大型网络上会引起多种问题。

首先，维护几百台机器上的**/etc/fstab** 文件就是很枯燥乏味的。而且每个文件都有点儿区别，需要逐个维护。

其次，如果是从许多不同主机上安装文件系统的，当其中有台服务器崩溃时，就会引发混乱。访问安装点的每条命令都会被挂起，不予执行。

第三，当一台重要的服务器崩溃时，会致使用户无法访问一些重要分区，如**/usr/share/man**。在这种情况下，如果能从备份服务器上暂时安装这些分区的一个副本，那就最好不过了。

自动安装守护进程在文件系统被访问到时就安装文件系统，而不再需要文件系统时，就卸装它们。这一过程减少了活动安装点的数目，而且对用户几乎是透明的。大多数自动安装程序还可以提供一系列“复制的”（等同的）文件系统，这样一来，当一台主服务器不能使用时，网络仍能继续工作。

为了完成幕后的安装和卸装功能，自动安装程序在每个被指定为自动安装的目录中安装一个虚拟文件系统驱动程序。过去，自动安装程序假装成 NFS 服务器来完成这一任务，但这种方法存在着若干明显的限制而很少在目前的系统中使用。现在使用的是一种驻留内核的文件系统驱动程序 autofs。

自动安装程序按照您在它的配置文件中列出的说明构成一个文件系统的层次结构来代替对实际的文件系统作镜像。当用户要访问自动安装程序的虚拟文件系统中的某个目录时，自动安装程序就会截获这次访问，并安装用户想访问的实际文件系统。

[5] 谈到喋喋不休的市场宣传用语，您经常会听到的主要词汇之一就是“网络附加存储（network attached storage）”，即 NAS。它只是“文件服务”的一种奇特说法而已。

www.TopSage.com

16.6.1 automount：根据需要安装文件系统

自动安装工具的思想最初来自于Sun公司。Linux的自动安装工具叫做autofs，虽然从概念上来说它是一个独立的实现，而且在若干方面都有不同，但其在设计上模仿了Sun公司的自动安装工具。

automount是一个后台进程，它为autofs配置一个安装点，autofs是Linux的自动安装工具在内核中的部分。启动脚本**/etc/init.d/autofs**分析一个master（主控）文件（通常是**/etc/auto.master**），并且为列出的每个安装点执行**automount**命令。一般会看到，已经配置的每个自动安装点都有一个正在运行的**automount**实例。

您很少需要直接运行**automount**命令，因为几乎所有对自动安装程序的管理工作都是通过**/etc/init.d/autofs**这个脚本来完成的[6]。和大多数启动脚本一样，**autofs**脚本可以在命令行接受一个参数，它可以是**start**、**stop**、**reload**、**restart**或者**status**。只要对自动安装工具的配置做了改动，您就必须运行**autofs reload**来让改动起作用。**autofs sttus**会提供给您有关现有**automount**的状态。

auto.master文件把一个安装点和一个“映射（map）”关联了起来。一个映射把被访问的目录名——称为“键（key）”——转换为**mount**可以用来执行实际安装操作的命令行。一个映射可以是一个文本文件、一个可执行程序或者一个NIS或者LDAP数据库。

当一个用户访问到已经用autofs内核文件系统模块安装的某个目录时，它就通知这次访问的用户级**automount**进程。**automount**进程通过参考一个映射文件或者程序，判断出要安装哪个文件，然后执行安装操作，再把控制权交还给启动这次查找的用户。

您可以通过运行**mount**和**ps**命令，看到autofs文件系统以及和它们关联的**automount**进程：

```
$ mount
/dev/hda3 on / type ext2 (rw)
proc on /proc type proc (rw)
/dev/hda1 on /boot type ext2 (rw)
automount(pid8359) on /misc type autofs          // automounter filesystem
    (rw,fd=5,pgrp=8359,minproto=2,maxproto=4)
automount(pid8372) on /net type autofs           // automounter filesystem
    (rw,fd=5,pgrp=8372,minproto=2,maxproto=4)

$ ps auxw | grep automount
root      8359  0.0  1.0  1360  652 ?        S    Dec27   0:00
    /usr/sbin/automount /misc file /etc/auto.misc
root      8372  0.0  1.0  1360  652 ?        S    Dec27   0:00
    /usr/sbin/automount /net program /etc/auto.net
```

从这里我们可以看到两个autofs文件系统，分别安装在**/misc**和**/net**上。这两个虚拟的文件系统分别关联到了PID为8359和8372的两个**automount**进程上。由**/etc/init.d/autofs**脚本运行的**automount**命令可以从**ps**的输出里看到。**auto.misc**是一个普通的映射文件，**auto.net**是一个可执行的程序。这两个映射都会在后面详细说明。

16.6.2 主控文件

主控（master）文件列出了应该有autofs文件系统安装到它们上面的目录，并且给每个目录关联了一个映射。除了给映射和映射名提供根目录之外，您还能以**mount**命令使用的“**-o**”格式来指定其他选项。这些选项会应用到映射中的每一项上。Linux的约定和Sun的不同，因为master文件的选项

[6] 不要把**autofs**脚本和autofs文件系统搞混了。它们之间的关系是，这个脚本告诉内核如何配置这种文件系统。

www.TopSage.com

是叠加到 map（映射）文件的选项之上的，两组选项都要交给 **mount**。

一个简单的主控文件类似于下面的样子，它利用了下面一节中给出的映射文件：

```
# Directory    Map             Options
/chimchim      /etc/auto.chim  -secure,hard,bg,intr
```

主控文件可以用一个通过 NIS 共享的版本来替换或者扩展。系统的 automount 信息的来源由 **/etc/nsswitch.conf** 文件中的 automount 域来指定。参考 17.4.3 节了解有关 **nsswitch.conf** 文件的更多信息。

16.6.3 映射文件

映射（map）文件（在其他系统上也叫做“间接映射”）自动把几个文件系统安装到一个公共的目录下。目录路径是在主控（master）文件中，而不是在映射文件本身里指定的。例如，安装在**/chimchim** 下的文件系统（对应于上面的例子）的映射文件可能看起来像是下面这样：

```
users    chimchim:/chimchim/users
devel    -soft,nfsproto=3 chimchim:/chimchim/devel
info     -ro  chimchim:/chimchim/info
```

第一列给出了子目录的名字，每个自动安装的目录都应该安装在这个目录下，后面的几项列出了安装点、文件系统的源路径。这个例子（保存在**/etc/auto.chim**）告诉 **automount**，它能从主机 **chimchim** 安装**/chimchim/users**、**/chimchim/devel** 和**/chimchim/info** 目录，**info** 为只读安装、**devel** 用 NFS 协议第 3 版以 soft（软）方式安装。

在这一配置中，chimchim 上的各个路径同本地主机上的各个路径是一样的。不过，并不要求一定有这种对应关系。

16.6.4 可执行的映射文件

如果一个间接映射文件是可执行的，它应该是个脚本或者程序，能动态地产生自动安装信息。代之以读取文本文件形式的映射关系，自动安装程序会带参数（“key”）来执行它，参数指出了用户试图访问哪一个子目录。脚本负责输出适用的映射项，如果指定的键（key）无效，那么脚本仅仅是退出而已，并不输出任何东西。

这项功能非常强大，它弥补了自动安装工具相当奇怪的配置系统中许多潜在的缺陷。实际上，它能让您很轻松地以您自己的格式定义一个适用全站范围的自动安装配置文件。您可以编写一个简单的 Perl 脚本来解读每台机器上的全局配置。有些系统自带一个**/etc/auto.net**，它是个方便的可执行映射文件，它把主机名作为键（key），安装那台主机上所有导出的文件系统。

值得一提的是，自动安装工具有一个容易混淆的特性：对于被自动安装的文件系统，当您要列出其父目录的内容时，无论那里已经自动安装了多少文件系统，该目录都显示为空。在图形用户界面的文件系统浏览器里，看不到自动安装的目录。例如：

```
$ ls /portal
$ ls /portal/photos
art_class_2004    florissant_1003            rmnp03
blizzard2003      frozen_dead_guy_Oct2004    rmnp_030806
boston021130      greenville.021129          steamboat2002
```

文件系统 **photos** 准确无误地自动安装在**/protal** 下。用完整的路径名能够访问到它。不过，查看**/protal** 目录却看不到它。如果是通过**/etc/fstab** 文件或者手工执行 **mount** 命令安装的这个文件系统，

www.TopSage.com

那么它就和其他任何目录一样，能看到是父目录下的一个子目录。

有一种变通方法，能解决上述看不到子目录的问题，这就是创建一个影子目录，其中包括链到自动安装点的符号链接。例如，如果**/automounts/photos** 是一个符号链接，它链到了**/portal/photos** 上，那么就可以用 **ls /automounts** 命令看到 **photos** 是一个自动安装的目录。访问**/automounts/photos** 仍然会经过自动安装工具处理，且访问结果正确。遗憾的是，需要维护这些符号链接，除非用一个脚本定期重新生成它们，否则会使之与实际的自动安装目录情况不能保持同步和一致。

16.7 推荐读物

CALLAGHAN, BRENT. *NFS Illustrated.* Reading, MA: Addison-Wesley. 1999.

STERN, HAL, MIKE EISLER, AND RICARDO LABIAGA. *Managing NFS and NIS (2nd Edition).* Sebastopol, CA:O'Reilly Media, 2001.

表 16.5 列出了用于 NFS 协议的各种 RFC。

表 16.5 和 NFS 有关的 RFC

RFC	标 题	作 者	日 期
1094	Network File System Protocol Specification （网络文件系统协议规范）	Sun Microsystems	1989 年 3 月
1813	NFS Version 3 Protocol Specification （NFS 第 3 版协议规范）	B. Callaghan 等	1995 年 6 月
2623	NFS Version 2 and Version 3 Security Issues （NFS 第 2 版和第 3 版的安全问题）	M. Eisler	1999 年 6 月
2624	NFS Version 4 Design Considerations （NFS 第 4 版设计的考虑事项）	S. Shepler	1999 年 6 月
3530	NFS Version 4 Protocol （NFS 第 4 版协议）	S.Shepler 等	2003 年 4 月

16.8 习题

★E16.1 研究您本地的 NFS 设置。使用了 NFS 吗？还是有别的不同解决方法？采用了自动安装方式吗？有什么样的权衡考虑？

★★E16.2 **mountd**、**nfsd** 和 **portmap** 之间有什么关系？NFS 对 **portmap** 的依赖在安全上意味着什么？

★★E16.3 NFS 作为一种无状态协议，其设计涉及哪些方面？特别地，讨论无状态对文件上锁、访问权限和安全方面产生的影响。有状态的网络文件系统和 NFS 会有什么样的不同？

★★E16.4 您的上司要您把**/usr** 和**/usr/local** 通过 NFS 导出。已知的信息和要求如下。

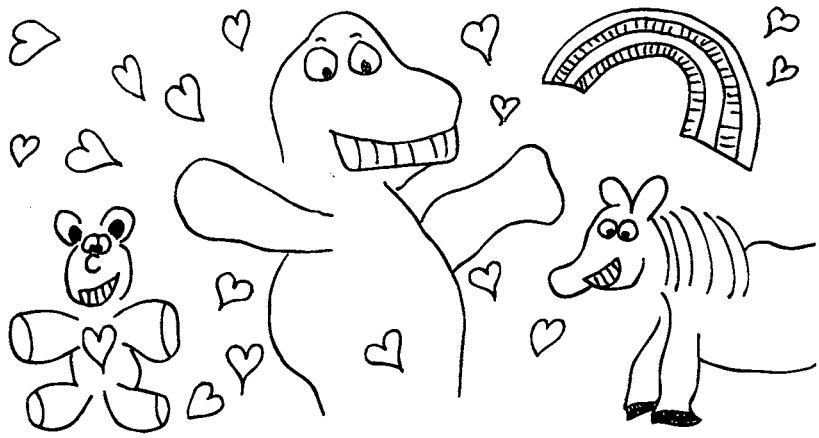
a）因为办公室的行政因素，您只想让您的部门（本地子网 192.168.123.0/24）能够使用这些被导出的文件系统。为了实现这种配置，必须要向什么文件加入什么样的配置行？注意给出正确的导出选项。

b）列出让 **mountd** 和 **nfsd** 认出这些新被共享的文件系统所需要的步骤。在不安装它们的情况下，您会怎样核实那些目录已经被共享出去了？

c）概述让您的本地子网使用**/mnt/usr** 和**/mnt/usr/local** 自动安装被导出的文件系统应采取的策略。

www.TopSage.com

第 17 章　共享系统文件



一个正常工作的系统要靠几十个甚至几百个配置文件都包含正确的信息。当您把一台主机上的配置文件数目乘以网络上的主机数目时，结果会是成千上万的文件——这么数目庞大的文件不可能由手工来配置。

在现实世界里，从系统管理的角度来看，计算机往往彼此类似。将多台机器组成共享配置信息的组，而不是在每台机器上编辑配置文件，这样做效率更高。当然，您可以按不同方式组合机器。

最简单的方法是，在一个地方保存每种配置文件的主控副本，一旦它有变动，就把它分发给组中的成员。这种解决方案的优点是简单，而且在每种 Linux（以及 UNIX）系统上都能工作。

另外一种方法是，把文本文件全部删掉，让每台机器从一台中央服务器获得它的配置信息。这比复制文件的方法要复杂，但它也能解决其他一些问题。例如，客户机不能错过更新，哪怕配置文件发生变化时这台客户机宕机也不能错过。而且，从服务器上获得信息也许比从文件要快，这取决于本地硬盘的速度和服务器高速缓存的大小。另一方面，当中央服务器崩溃时，整个网络就会暂停。

让情况更难处理的是，现在大多数机构都面临要支持多种平台的混合——有些机器是 UNIX，有些是 Linux，而有些是 Windows——当用户不得不承受由此带来的诸多不便时，比如不得不记住（和修改）每种平台上的不同口令，他们的抱怨会与日俱增。在性质迥异的系统（比如 Windows 和 Linux）之间保持配置和用户信息的同步一致，曾经是那么可望不可及。现如今却变得垂手可得。

开发用于大型网络的分布式管理数据库的工作历史能追溯到过去几十年，而且已经产生了一系列令人感兴趣的系统。然而，就总体而言，这些系统目前似乎在其解决方法上都还没有做到位。有些系统虽然简单但不安全且没有扩展性。有些系统虽然功能不错但不易使用。所有这些系统都有局限性，不能按照人们想要的方式来配置网络，它们中间也没有哪一种能管好您想要在多台主机之间共享的所有信息。

www.TopSage.com

本章先讨论一些在网络上保持文件同步的基本技术。然后介绍 NIS，它最初为 UNIX 引入，过去曾是最流行的管理数据库系统。最后，我们介绍 LDAP，它是一种更先进且与平台无关的系统，正在成为一种事实上的标准。现如今的大型网点正在向 LDAP 迁移，其推动力主要因为微软在自己的 Active Directory 产品中采用了 LDAP 标准，还有就是人们希望更好地集成 Linux 和 Windows 环境。

17.1 共享什么

在一个 Linux 系统上许许多多的配置文件中间，只有一部分可以在多台机器之间共享使用。表 17.1 列出了最常见的共享文件。

表 17.1 常被共享的系统文件

文件名	功能
/etc/passwd	用户账号信息数据库
/etc/shadow[a]	保存隐蔽的口令
/etc/group	UNIX 组定义
/etc/hosts	主机名和 IP 地址之间的映射关系
/etc/networks[b]	文字名与 IP 网络号的关联
/etc/services	列出众所周知的网络服务的端口号
/etc/protocols	将文本名字映射到协议号
/etc/ethers[b]	主机名和以太网地址之间的映射
/etc/mail/aliases	包含电子邮件别名
/etc/rpc	列出 RPC 服务的 ID 号
/etc/netgroup[b]	定义主机、用户和网络的集合
/etc/cups/printcap	打印机信息数据库
/etc/printcap.cups	打印机信息数据库（另一个路径）
/etc/termcap	终端类型信息数据库

a. 没有必要能和其他风格的 UNIX 共享，因为加密的类型可能不同，参见 6.1 节。
b. 未被所有系统采用。

其他许多配置文件也有可能在多个系统之间共享，这取决于您想要让您网点上的机器有多大的相似程度。对于大多数网点来说，其他配置文件都和特定的应用（例如，**sendmail** 的配置文件 **/etc/sendmail.conf**）相关，像 NIS 和 LDAP 这样的管理数据库系统不支持它们，您必须通过复制的方式来共享这些文件。

以前，表 17.1 中列出的文件里有许多都是通过标准 C 库中的例程来访问的。例如，可以用例程 **getpwuid**、**getpwnam** 和 **getpwent** 来搜索 **/etc/passwd** 文件。这 3 个例程能够打开、读取和分析 **passwd** 文件，这样一来，用户级程序就不必自己去做这些事情。现代的 Linux 发行版本也使用了 PAM（Pluggable Authentication Module，可插入式身份验证模块），它为执行与安全相关的查找操作提供了标准的编程接口。PAM 模块可以让 Kerberos 这样的身份验证系统很容易地集成到 Linux 里面。参考 20.5.6 节了解有关 PAM 的更多信息。

管理数据库系统为这些信息中的很多都提供了另一种来源，这使得情况更复杂了。除了标准的纯文件之外，传统的 C 库例程（**getpwent** 等）还能理解常见的数据库系统，并且能访问它们。系统管理员决定到底要补充哪些数据源供查询，参考 17.4.3 节了解更多的信息。

www.TopSage.com

17.2 nscd：缓存查找的结果

在有些发行版本上，影响系统文件的另一个因素是 **nscd**，即名字服务缓存守护进程（name service cache daemon），这个名字起的有点儿误导性。**nscd** 和 C 库配合起来就能缓存像 **getpwent** 这样的库调用的结果。**nscd** 只是对这些库例程的一个封装程序（wrapper），它对于实际要查的数据源一无所知。从理论上讲，**nscd** 应该能提高查找的性能，但是从用户的主观角度来看，基本上感觉不到有什么提高。

“名字服务缓存守护进程”的说法可能有误导性，因为术语“name service（名字服务）”一般是指 DNS 这种在主机名和 Internet 地址之间建立映射关系的分布式数据库系统。**nscd** 实际上确实能缓存 DNS 的查询结果（因为它封装了 **gethostbyname** 等库例程），但是它也封装了其他库例程，能访问 **passwd** 和 **group** 文件及其网络数据库的等价物中保存的信息（出于安全原因，不缓存对**/etc/shadow** 文件的查找结果）。参考第 15 章了解有关 DNS 的更多知识。

从概念上讲，除了加快查找速度之外，**nscd** 应该对系统的运行没有影响。但在实际中间，它会造成意想不到的效果，因为它保留了它自己的一份查找结果副本。查找结果都要在缓存里保存一段固定长度的时间（在 **nscd** 的配置文件**/etc/nscd.conf** 里设置），而且一定有可能出现这样的情况，就是在以前的数据没有超时作废之前，最近的变动不会反映到 **nscd** 的缓存里。**nscd** 还是够聪明的，它能监视本地数据资源（比如**/etc/passwd**）是否变化，所以本地的变化应该能在 15 秒内公布出来。对于远程的数据源，比如通过 NIS 来检索的数据来说，在变动生效之前，您可能不得不等上一段时间让缓存过期。

在我们举例的发行版本中，只有 SUSE 默认运行 **nscd**。Fedora 和 RHEL 虽然安装了 **nscd**，但是在默认情况下不会在系统引导的时候启动它，要想使用 **nscd**，只需运行 **chkconfig nscd on** 即可。Debian 和 Ubuntu 都兼容 **nscd**，但它们的默认安装里并没有带 **nscd**，可执行 **apt-get install nscd** 下载它。

nscd 在系统引导时启动，并连续运行。默认的**/etc/nscd.conf** 文件指定的 **passwd** 数据过期时间为 10 分钟，**hosts** 和 **group** 文件的过期时间为 1 小时，对于没有找到的结果，过期时间为 20 秒（在重新尝试一次没有成功的查找之前要等的时间）。在实际中，很少需要调整这些值。如果您发现自己想知道，为什么您最近所做的一次改动似乎并没有起作用，可能就是 **nscd** 的原因。

17.3 把文件复制到各处

虽然竭尽全力到处复制文件不是一种妥善的方法，但它可以在各种类型的机器上采用，而且设置和维护都很简单。另外，由于该方法把机器之间的相互依赖性减至最小（虽然它也更容易让这些机器出现不一致的情况），所以也是一种可靠的系统。从什么可以散布，如何散布这个方面来讲，复制文件的做法灵活性也最强。实际上，除了系统文件之外，它也经常用来保持应用和数据文件的更新。

常见的数据库服务都不支持的配置文件相当少（这类文件的一些例子有：决定 **sendmail** 如何传递邮件的**/etc/sendmail.cf**，决定主机如何保持它们的时钟同步的**/etc/ntp.conf**）。如果您确实要保持这类文件的同步（这样做通常是明智的），即使您也用了 NIS 或者 LDAP 分发其他类型的配置信息，但除了求助某种到处复制文件的系统之外，您确实别无选择。

文件复制系统既可使用一种“推（push）”模型，也可使用一种“拉（pull）”模型。在推模型中，主服务器定期分发最新文件给每个客户机，而不管客户机是否需要这些文件。只要文件发生变化，就可明确地推送文件，或者按照一种常规进度（也可能有些文件比其他文件传送得频繁些）分发该文件。

推模型的优点是把分发系统集中放在一台服务器上。文件、客户机列表、更新脚本以及时间表都保存在一个地方，有利于对实施方案进行控制。它的一个缺点是，每台客户机必须允许服务器能修改

www.TopSage.com

它的系统文件，这就带来了一种安全风险。

在拉模型的系统中，每台客户机负责从服务器更新它自己的信息。这样一来，分发文件不再那么集中，但这种方法适应性更强，而且更安全。当数据跨越管理分界进行共享时，这种拉系统特别有吸引力，因为主服务器和客户机不必由同一个管理团队或者行政部门运行。

17.3.1 rdist：推文件

rdist 命令是从一台中央服务器分发文件的最容易的途径。它与 **make** 命令的功能有些类似：使用文本编辑器创建一个对要分发的文件进行说明的规范，然后执行 **rdist** 命令实现该规范。由于只有在文件过期时 **rdist** 才复制文件，所以，您在写规范说明时就当所有文件都将被复制一样，让 **rdist** 自己优化掉不必要的工作即可。

rdist 保留文件的属主、属组、模式和修改时间。当 **rdist** 更新一个已有的文件时，它首先删除旧版本，然后再安装新版本。这一特性让 **rdist** 适合于传输在更新期间有可能用到的可执行文件[1]。

rdist 命令传统上是在 **rsh** 上运行的，依靠 **rsh** 方式的身份验证来访问远程系统。遗憾的是，这种系统不安全，在现代操作系统上默认被禁用。即使 **rdist** 的文档还继续在说 **rsh**，但不要误以为 **rsh** 是一种明智的选择。

当前版本的 **rdist** 要好一些，因为它们会用其他任何能理解相同语法的命令来替换 **rsh**。实际上，代替它的是 **ssh**，**ssh** 采用加密方式来验证主机的身份，并防止网络窃听者获得您的数据的副本。不便之处是您必须让远端的 **ssh** 服务运行在一种不需要口令的模式下（但是验证客户机的身份要用一个加密的密钥对）。比起我们的一般建议来说，这是一种不太安全的配置，但是和 **rsh** 相比，它仍然是巨大的改进。参见 20.11.3 节了解有关 **sshd** 及其身份验证模式的信息。

我们已经阐述了执行 **rdist** 带来的危险，现在我们来看看它实际上是如何工作的。与 **make** 类似，**rdist** 在当前目录下寻找一个控制文件（**Distfile** 或 **distfile**）。**rdist -f** *distfile* 明确地给出了控制文件的路径名。在 **Distfile** 文件中，制表符、空格和换行符均可作为分隔符互换使用。以符号#开头的行是注释行。

Distfile 文件的主体主要由类似于下面形式的语句组成。

```
label: pathnames -> destinations commands
```

lable 字段把一个名字和语句关联起来。您可以从 shell 执行 **rdist** *label*，只分发在一条特定语句中描述的文件。

pathnames 和 *destinations* 分别是要被复制的文件列表，以及它们的目的主机。如果列表不只一项，则必须用括号括起来，并且各部分之间用空白隔开。*pathnames* 可以包含 shell 风格的通配符（如**/usr/man/man[123]** 或**/usr/lib/***）。符号~user 也可接受，但在源主机和目的主机上具体表示什么目录则相互无关。

默认情况下，**rdist** 把 *pathnames* 列出的文件和目录复制到每台目的机器上的等价路径下。您可以通过提供一系列命令来改变这种做法。每条命令以分号结束。

它能够理解下面几条命令：

install *options* [*destdir*];
notify *namelist*;
except *pathlist*;
except_pat *patternlist*;
special [*pathlist*] *string*;
cmdspecial [*pathlist*] *string*;

[1] 虽然文件的旧版本已经从文件系统的名字空间中消失了，但它继续存在，直到所有的访问都结束为止。在管理日志文件时，也必须注意到这种影响。参考 10.1.3 节了解更多的信息。

www.TopSage.com

install 命令设置影响 **rdist** 复制文件方式的选项。这些选项主要控制对符号链接的处理、**rdist** 差别检查算法的纠正以及删除的处理方式。这些选项应该以-o 开头，并且由一组以逗号分隔的选项名所组成。例如，下面这行：

```
install -oremove,follow ;
```

让 **destdir** 沿着符号链接复制（而不是只复制链接本身）文件，并且去除在目的主机上已经有，而在源主机上又没有对应文件的文件。参考 **rdisk** 的手册页了解全部选项。默认的方式几乎总能满足您的需要。

“install”这个名字有点误导性，因为不管有无 install 命令都会复制文件。在 **rdist** 命令行上，可以根据需要随时指定选项，但是在 **Distfile** 文件中包含选项的时候，它们就只适用于由 intall 命令来处理的文件集合。

选项 *destdir* 指定了在目的主机上的一个安装目录。默认情况下，**rdist** 使用原来的路径名。

notify 命令的参数是一个电子邮件地址的列表。只要有一个文件被更新，**rdist** 便向这些地址发送邮件。对于任何不包含@的地址，rdist 将在其后加上目的主机名。例如，当 **rdist** 要把一个更新文件列表发送给主机 anchor 时，会先将“pete”扩展成 pete@anchor 。

except 和 except_pat 命令从要复制文件的列表中删除路径名。except 的参数直接照字面进行匹配，而 except_pat 的参数按规则表达式进行解释。这两个 except 类命令很有用处，因为与 **make** 类似，**rdist** 允许在其控制文件的开头部分定义宏。如果若干个语句使用的文件列表都相似，则只需为每台主机指出要添加或删除的部分即可。

special 命令在每台远程主机上执行一条 shell 的命令（命令用引号括起来的参数 *string* 指定）。如果出现 *pathlist*，那么 **rdist** 就在每复制一个指定的文件后，执行一次这条命令。如果没有 *pathlist*，**rdist** 就在每个文件之后执行这条命令。遗憾的是不能只在复制完所有文件后，执行一次命令。cmdspecial 命令与此类似，但它在复制完成之后仅执行一次那条 shell 命令（*pathlist* 的内容作为一个环境变量被传给 shell）。

下面给出一个 Distfile 的简单例子：

```
SYS_FILES  = (/etc/passwd /etc/group /etc/mail/aliases)
GET_ALL    = (chimchim lollipop barkadon)
GET_SOME = (whammo spiff)

all: ${SYS_FILES} -> ${GET_ALL}
    notify barb;
    special /etc/mail/aliases "/usr/bin/newaliases";

some: ${SYS_FILES} -> ${GET_SOME}
    except /etc/mail/aliases;
    notify eddie@spiff;
```

这个配置文件复制主机 chimchim、lollipop 和 barkadon 上所列的 3 个系统文件，并且一旦出现更新或错误，即向 barb@*destination* 发送邮件。复制完**/etc/mail/aliases** 文件后，**rdist** 会在每台目的主机上运行 **newaliases**。向机器 whammo 和 spiff 上复制的文件只有两个，而且不运行 **newaliases**，并向 eddie@spiff 发送报告。有关 **newaliases** 的更多信息，请参见 18.4.6 节。

要让 **rdist** 在多台机器间工作，您也必须让接收方主机上的 **sshd** 信任您正从上面分发文件的主机。要做到这一点，您可以为主控主机生成一个纯文本的密钥，并且在每台接收方主机上的 **~root/.ssh/authorized_keys** 文件中保存密钥的公开部分的一个副本。限制这个密钥能够做的事情，并且限制能从哪里登录，这样做可能比较明智。参考 20.11.3 节了解更多的信息。

17.3.2 rsync：更安全地传输文件

rsync 是由 Andrew Tridgell 和 Paul Mackerras 编写的，其思想与 **rdist** 类似，但侧重点有些不同。对于

www.TopSage.com

“文件复制过程”的控制文件，**rsync** 使用这个文件的方式和 **rdist** 的方式不同（虽然服务器端确实有一个配置文件）。**rsync** 有点儿像是能严格保留链接信息、修改时间和权限的 **rcp** 的改进版。**rsync** 在网络上的执行效率比 **rdist** 更高，因为它能读取每个文件的内容，并尝试只传输各版本间有差异的部分。大多数 Linux 发行版本都提供了 **rsync**，虽然它可能不是默认安装的软件。从 rsync.samba.org 可以得到 **rsync**。

从我们的观点来看，**rsync** 的主要优点是，接收文件的机器可以将 **rsync** 的远端作为 **xinetd** 或者 **inetd** 的一个服务器进程运行。服务器（实际上是另外一种模式的 **rsync**，必须在主控机器和客户机上都安装）的可配置性很强：它可以限制远程只能访问一部分目录，并且要求主控机器提供口令验证其身份。因为不必用 **rsh** 访问，所以设置 **rsync** 来分发系统文件不会引起太多的安全问题（然而，如果更喜欢用 **ssh**，而不用基于 **inetd** 的服务器进程，**rsync** 也允许这么做）。另外，**rsync** 也能以拉模式运行（从 **rsync** 服务器上把文件拉下来，而不是把它们推到客户机上），这种模式更安全（参考 17.3.3 节有关拉文件的内容）。

遗憾的是，**rsync** 没有 **rdist** 那么好的灵活性，它的配置要比 **rdist** 的 **distfile** 简单。您不可以在客户机上随意执行命令，也不能一次 **rsync** 到多台主机上。

下面这条命令就是一个例子：

```
# rsync -gopt --password-file=/etc/rsync.pwd  /etc/passwd lollipop::sysfiles
```

它把**/etc/passwd** 文件传送给 lollipop 这台机器。选项**-gopt** 保留了这个文件的权限、所有权关系以及修改时间。在 **lollipop::sysfiles** 中间的两个冒号让 **rsync** 在 873 端口上直接和远程的 **rsync** 联络，而没有用 **ssh**。保存在**/etc/rsync.pwd** 中的口令对连接进行验证[2]。

这个例子只传一个文件，但是 **rsync** 能一次处理多个文件。此外，**--include** 和**--exclude** 两个标志还能让您指定一系列匹配文件名的正则表达式，这样您就可以建立起一套相当复杂的传送规则。如果命令行用起来太不方便，您也可以用**--include-file** 和**--exclude-file** 选项从独立的文件中读取到匹配模式。



只要装好 **rysnc** 的软件包，那么 Red Hat、Fedora 和 SUSE 都给 **rsync** 提供了 **xinetd** 的配置。不过要编辑文件**/etc/xinetd.d/rsync**，把 `disable=yes` 这行改为 `disable=no`，切实启动服务器。



Debian 和 Ubuntu 的 **rsync** 软件包要在**/etc/inetd.conf** 里加入一行配置，才能启动 **rsync** 服务器的功能。假如使用 TCP wrappers，那么加入的这一行如下：

```
rsync  stream tcp nowait  root /usr/sbin/tcpd  /usr/bin/rsyncd  --daemon
```

如果您用到了 TCP wrappers，那么可能想要配置 **tcpd** 来封锁除了分发系统文件的那台机器之外所有主机的访问。在 **rsyncd.conf** 中也可以指定拒绝主机的功能，但树立多道屏障始终不是坏事。

一旦您已经启用了 **rsync**，就需要设置两个配置文件来告诉 **rsync** 服务器如何行事。主要的文件是**/etc/rsyncd.conf**，它既包含有全局的配置参数，又包含一组“模块”，每个模块都是一个导出或者导入的目录树。用于一个您可以推送的模块（也就是说，它会接受由连接的客户机发起传入的文件）的合理配置看上去就像下面这样：

```
# sysfiles is just an arbitrary title for the particular module.
[sysfiles]
# This is the path you allow files to be pushed to. It could be just /.
path = /etc
# This is the file specifying the user/password pair to authenticate the module
secrets file = /etc/rsyncd.secrets
```

[2] 虽然口令在网络上不会以明文发送，但是传输的文件并没有被加密。如果使用 ssh 进行传输（即 **rsync -gopt -e ssh /etc/passwd/ etc/shadow lollipop:/etc**，注意只有一个冒号），则连接会被加密，但是必须配置 **sshd** 不要求口令。由您选择用哪种方法！

www.TopSage.com

```
# Can be read only if you are pulling files
read only = false
# UID and GID under which the transfer will be done
uid = root
gid = root
# List of hosts that are allowed to connect
hosts allow = distribution_master_hostname
```

其他许多选项都可以设置，但是默认的设置已经相当合理了。这个配置将操作限制在了**/etc** 目录上，而且只允许被列出的主机能访问它。从用户或者说客户机的角度来看，您可以用形如 *hostname*::**sysfiles** 这样的目的地址(它映射到前面的模块)，把文件 **rsync** 到服务器，如果您想要将 **rsync** 设为拉模式（从一台中央 **rsync** 服务器拉回文件），上述的配置仍然能起作用，即便您可能想要让某些地方更严密一点儿（例如，设置传输模式为只读）。

您需要做的最后一件事情就是设置一个 **rsyncd.secrets** 文件。它一般保存在**/etc**（虽然您可以把它放到别处去）下，包含有客户机用于验证自己身份的口令。例如：

```
root:password
```

一般而言，**rsync** 的口令应该和用户实际的口令不一样。因为 **rsync** 的口令是以明文出现的，**rsyncd.secretes** 必须只能由 root 读取。

17.3.3 拉文件

实现一个拉系统可以有几种方法。最直接也是最简单的方法就是能够从一台中央的 ftp 或者 Web 服务器[3]得到文件，而且让客户机根据需要自动下载这些文件。在历史上，系统管理员都是用自己的工具程序来做这件事的（经常是写某种脚本来做 **ftp**，比如用 **expect** 写），但是现如今已经有标准的工具程序能替您做这件事情了。

大多数 Linux 的发行版本都带的一种流行的工具程序就是 **wget**。它是一种小程序，用法很直观，能取得一个 URL（要么是 FTP，要么是 HTTP）下的内容。例如，用 **wget** 来 FTP 一个文件的话，只需运行：

```
wget ftp://user:password@hostname/path/to/file
```

被指定的 *file* 就会被下载到当前目录下。

另一种方法只能用于 FTP，它就是 **ncftp**，大多数 Linux 的发行版本也带这个程序。它实际上只是一种功能增强的 FTP 客户端，便于在脚本中使用。

有些站点分发文件的方式是在像 NFS 这样的网络文件系统上公布文件。从自动的角度来看，这或许是最简单的一种技术——需要做的全部工作就是 **cp**，至少从理论上讲是如此。在实际中，您可能想要有点儿复杂的功能，在盲目地复制一个系统文件之前，检查是否出现了有安全问题的迹象，以及文件内容是否受损。从安全的角度来看，在 NFS 上公布敏感的系统文件有若干缺点，但它是一种简单且有效的转移数据的方式。

最后一种选择是使用上一节介绍的 **rsync**。如果您在中央分发主机上运行了 **rsync** 服务器，那么客户机只要用 **rsync** 把文件下载下来就可以了。这种方法比起用 FTP 来，可能稍微复杂了一些，但是它能让您用上 **rsync** 的全部功能。

无论您使用的是什么系统，都要注意不要让您的数据服务器负载过重。如果网络上有大量的主机在同时访问您的服务器（例如，如果每台客户机都在同一时刻通过 **cron** 执行更新操作），您可能在无

[3] 要记住，HTTP 和 FTP 都以明文方式传输数据。如果要顾及到安全性，那么可以分别考虑 HTTPS 或者 SFTP。

www.TopSage.com

意中造成了一次拒绝服务攻击。大型的网点应该注意这个问题，让更新时间交错或者随机。做到这一点的一种简单方法就是像下面这样，用 Perl 脚本把 **cron** 任务封装起来：

```
#!/usr/bin/perl
sleep rand() * 600; # sleep between 0 and 600 seconds (ie, 10 minutes)
system(copy_files_down);
```

17.4 NIS：网络信息服务

NIS 于 20 世纪 80 年代由 Sun 公司首次发布，它是第一个“成熟的”管理数据库。它最初叫做 Sun Yellow Pages，但最终因法律原因改名为 NIS。NIS 命令仍然以字母 **yp** 开头，看来它难以忘却原来的名字。NIS 得到了 UNIX 厂商的广泛支持，每种 Linux 发行版本都支持它。

NIS 的共享单元是记录，不是文件。一条记录通常对应配置文件中的一行。主控服务器维护着系统文件的权威副本，这些系统文件以原来的格式保存在原来的位置上，之前只能用文本编辑器对其进行编辑。服务进程让这些文件的内容可通过网络进行访问。一个服务器和它的客户机就构成一个 NIS “域”[4]。

数据文件由 Berkeley DB 散列库（**gdbm**）预先处理成数据库文件，这一处理过程提高了查找效率。在主控服务器上编辑好文件后，您可以告诉 NIS 用 **make** 将它们转换成散列格式。

每项只允许关联一个关键字，于是一个系统文件可能不得不要转换成几个 NIS“映射文件”。例如，**/etc/passwd** 文件被转换成两个不同的映射文件 **passwd.byname** 和 **passwd.byuid**。前者用于通过用户名查找项，后者用于通过 UID 号查找项。两个文件都可用来列举出 **passwd** 文件中的项。不过，由于散列库不保留记录的顺序，所以没有办法精确地重建原来的文件（除非文件排过序）。

NIS 允许在一组从服务器上复制网络映射文件。提供一个以上的服务器可以减轻主控服务器的负担，即便在某些服务器不能用时，客户机仍能正常运行。每当主控服务器上的一个文件发生变化时，相应的 NIS 映射文件必须被推送到从服务器，这样才能保证所有服务器提供的数据都相同。客户机并不区分主服务器和从服务器。

传统的 NIS 实现要求每个物理网络至少放置一台 NIS 服务器。客户机使用 IP 广播方式定位服务器，而路由器和网关不转发广播包。**ypset** 命令可用来把一个客户机指向一个服务器，然而，一出现有问题的征兆，客户机会通过广播方式尝试定位一个新的服务器。除非客户机所在网络有一台服务器给予响应，否则上述连续动作会导致客户机挂起。

这种系统造成了许多问题，其中一个不小的问题是它极不安全。入侵者可以建立一个恶意的 NIS 服务器来响应广播，不是提供伪造的数据，就是限制并阻塞实际的请求，造成拒绝服务攻击。现如今，首选的管理技术是提供给每台客户机一张它自己合法的 NIS 服务器的明确清单。这种系统也有优点，那就是服务器不需要在本地子网上。

在 Linux 下，服务器在**/etc/yp.conf** 文件里列出。这里是一个例子：

```
domain    atrustnis    server    10.2.2.3
domain    atrustnis    server    10.2.2.4
```

每台服务器用一行表示，如果一台服务器宕机，NIS 会做故障切换，换到另一台服务器上。注意要以 IP 地址的形式列出服务器。**yp.conf** 接受主机名，但这些主机名必须在系统引导的时候能解析出来（例如，在**/etc/hosts** 文件中一一列出，或者能够通过 DNS 解析出来）。

如果必须使用广播模式，则语法为：

[4] 不要把 NIS 域与 DNS 域混淆。它们是完全独立的，彼此之间没有任何关系。

www.TopSage.com

```
domain    atrustnis    broadcast
```

17.4.1 NIS 的工作方式

NIS 的数据文件都保存在**/var/yp** 这个目录下。每个 NIS 映射文件都以散列格式存于这个 NIS 域的 NIS 目录的子目录下。每个关键字一个映射（文件），用它来搜索文件。例如，在域“cssuns”中，**/etc/passwd** 映射文件的 DB 文件可以是：

```
/var/yp/cssuns/passwd.byname
/var/yp/cssuns/passwd.byuid
```

passwd 文件可以用名字和 UID 两个关键字来搜索，所以从它派生出了两个映射文件。

makedbm 命令从纯文件生成 NIS 映射文件，不过，您不必直接调用这条命令。**/var/yp** 下的一个 **Makefile** 会生成所有常见的 NIS 映射文件。在您修改了一个系统文件之后，可用 **cd** 进入到**/var/yp** 下并运行 **make**。**make** 会把每个文件的修改时间和该文件派生出的映射文件的修改时间进行比较，并运行 **makedbm**，产生需要重建的映射文件。

ypxfr 命令将映射文件从主服务器复制到从服务器。**ypxfr** 是一个“拉”命令，它必须在每个从服务器上运行，好让该服务器导入映射文件。从服务器经常频繁执行 **ypxfr** 命令，这只是为了确保它们拥有最新的映射文件，你可以用 **cron** 控制服务器执行 **ypxfr** 的频率。

实现映射文件复制的默认工具效率有点低。Linux 提供了一种叫做 **ypxfrd** 的守护程序，它能在主服务器上运行，以更快的方式响应 **ypxfr** 请求。**ypxfrd** 绕过正常的 NIS 协议，只简单地分发映射文件的副本。

yppush 是一个“推”命令，它在主服务器上使用。它实际上不传输任何数据，而是指导每个从服务器执行 **ypxfr**。可通过 NIS 目录下的 **Makefile** 来使用 **yppush** 命令，以确保更新后的映射文件被传播到从服务器。

ypservers 是一个特殊的映射文件，它不对应任何纯文件。这个映射文件包含域中所有服务器的一个列表。当使用 **ypinit**（参见 17.4.5 节的内容）设置域时，就会自动产生该映射文件。只要主服务器需要向从服务器分发映射文件，主服务器就会考察 **ypservers** 文件的内容。

在初始配置之后，NIS 系统只有两个动态的组成部分：**ypserv** 守护进程和 **ypbind** 守护进程。守护进程 **ypserv** 仅在服务器上（主服务器和从服务器）运行，它接受来自客户机的查询，通过查找散列映射文件中的信息来回答查询。

ypbind 在 NIS 域中包括服务器在内的每台机器上运行。每当 C 库需要回答一次管理查询时，它就会连络本地的 **ypbind** 守护进程（假如**/etc/nsswitch.conf** 要求这么做）。**Ypbind** 在适当的域中找出一个 **ypserv**，并把它的身份返回给 C 库，而 C 库再直接联络那台服务器。这一查询机制如图 17.1 所示。

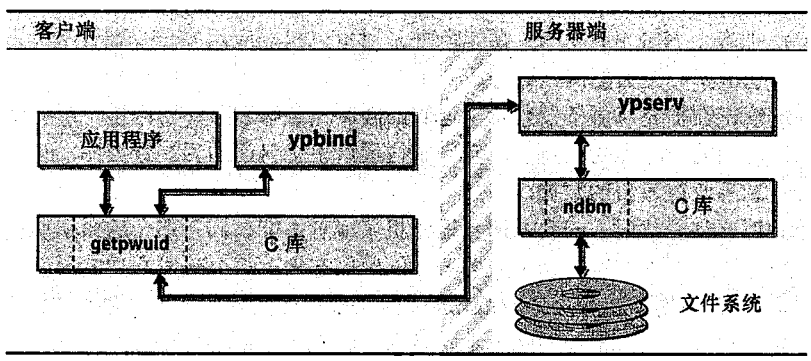


图 17.1 NIS 查询的过程

www.TopSage.com

当前 Linux 上 **ypbind** 的版本会定期检查，保证它们正在用的是一个 NIS 域内响应速度最快的服务器。这是对传统实现的一个改进，过去的办法都是固定使用一个特定的服务器。另一个只有在 Linux 上才有的特性是，客户机可以把不同的 NIS 域绑定到不同的映射文件上。

NIS 包括若干小命令，用于检查映射文件、找出每台服务器使用的映射文件的版本，以及控制客户机与服务器之间的绑定关系。NIS 命令和守护进程的一个完整清单见表 17.2。

表 17.2 NIS 命令和守护进程

程 序	描 述
ypserv	NIS 服务器守护进程，在系统引导时启动
ypbind	NIS 客户机守护进程，在系统引导时启动
domainname	设置机器所在的 NIS 域（在系统引导时运行）
ypxfr	从主服务器下载映射文件的当前版本
ypxfrd	响应从 **ypxfr** 来的请求（在主服务器上运行）
yppush	让从服务器更新其映射文件的版本
makedbm	从纯文件构建散列的映射文件
ypmake	从修改后的纯文件重建散列的映射文件
ypinit	将主机配置成主服务器或从服务器
ypset	使 **ypbind** 连接到特定服务器[a]
ypwhich	找出当前主机使用的服务器
yppoll	找出服务器正在使用的映射文件的版本
ypcat	输出 NIS 映射文件中的值
ypmatch	输出映射文件中特定关键字的项
yppasswd	改变 NIS 主服务器的口令
ypchfn	改变 NIS 主服务器的 GECOS 信息
ypchsh	改变 NIS 主服务器的登录 shell
yppasswdd	**yppasswd**、**ypchsh** 和 **ypchfn** 的服务器

a. 必须专门用 **ypbind -ypsetme** 或者 **ypbind-ypset**（危险）启用。

17.4.2 NIS 的优缺点

NIS 的优点之一是容易被人理解。NIS 可以比作是把文件到处复制，在大多数场合，管理员都不需了解 NIS 的内部数据格式。管理仍旧是使用原来同样的纯文件来完成的，只是要学一两个新的处理过程罢了。

因为没有办法把多个 NIS 域连起来，所以除非是对每台机器应用同一个配置文件，否则 NIS 不适合管理大型网络。您可将一个大型网络划分成几个 NIS 域，但每个域必须单独管理。即使一个大型网络确实就用一种配置，在扩充从服务器数量上遇到的限制也意味着，在实际中，这些站点通常要有别的一些机制来保证它们的 NIS 服务器同步。它们往往最后用了自己的后端数据库，而让它们的 NIS 服务器从这一中央信息源取得各自的数据。

如果在某个映射文件变化时，一台从服务器停机了或者不能访问，那么这台从服务器上的副本就不会被更新。从服务器必须周期地轮询主服务器，以确保它们有每个映射文件的最新版本。虽然 NIS 提供了轮询的基本工具，但您仍然必须使用 **cron** 才能实现想要的轮询方案。即便如此，仍然有可能在某段时间内同时提供同一个映射文件的两个不同版本，而客户机则随机得到其中一个。关于 **cron** 的详情参看第 8 章。

www.TopSage.com

NIS 安全性很低。广播模式尤其糟糕，网络上任何主机都可以声称为某个域提供服务，从而将假的管理数据发送给 NIS 客户机。您可以为每台客户机明确举出允许使用的 NIS 服务器，以此避免这个问题。

您可以在**/var/yp/securenets** 中明确地列出能读取服务器上的映射文件的主机来进行限制，不过，这一技术并非 100%安全。您还可以用一些别的技术（比如 **rdist** 或者 **rsync**）分发隐蔽的口令文件（shadow），以此加强安全性，我们不建议使用 NIS 来提供隐蔽口令文件的服务。

Linux 上比较老的 NIS 版本都有已知的安全漏洞。如果您运行了一个比较老的系统，那么在启动 NIS 之前要获得最新的补丁更新。

17.4.3 制订管理信息源的优先级

配置信息能以几种途径来分发。每个系统都理解纯文件，也知道怎样使用 DNS 来查找主机名和 Internet 地址。大多数系统还能理解 NIS。对于某种给定的信息可能有好几种潜在的来源，所以 Linux 提供了一种途径，能让您指定要检查的信息源以及检查的顺序。

在 NIS 最初的实现（在 Linux 之前）中，有些配置文件（特别是**/etc/passwd** 和**/etc/group** 文件）必须配置成“邀请”进入相应的 NIS 映射文件的内容。通过在文件本身中包含特殊符号可以把这种邀请方式进行扩展。位于一行开头的单独一个“+”号将包括整个 NIS 映射文件，“+@netgroup”将只包括与给定的网络组相关的项，“+name”则只包括一项。

这种方法并不好，大多数系统采用一个统一的配置文件**/etc/nsswitch.conf** 来代替它，这个文件为每种类型的管理信息指明一条搜索路径。使用一种兼容模式可以模拟出原来的做法，但是您不太可能会实际执行这种模拟（遗憾的是，模拟是大多数发行版本的默认做法）。

下面给出一个典型的 **nsswitch.conf** 文件的例子：

```
passwd: files nis
hosts:  files dns
group:  files
...
```

每行配置一种类型的信息（通常相当于一个纯文件）。常见的信息源有 `nis`、`nisplus`、`files`、`dns` 和 `compat`，它们分别指 NIS、NIS+[5]、普通纯文件（这类文件忽略“+”这样的记号）、DNS 以及 NIS 化的纯文件（有“+”）。DNS 是唯一一种提供主机和网络信息的有效数据源。

一个共享库（**/lib/libnss***）支持一种信息源，所以各个发行版本在它们所支持的信息源上略有不同。有些发行版本支持外部的 LDAP（参见 17.5 节）或者 Hesiod，后者是一种基于 DNS 的目录服务。Linux 上支持的另一种常见信息源（遗憾的是，文档不好）是 db，它从**/var/db** 读取映射文件（例如，**/var/db/passwd.db**）的一个散列处理过的版本。如果您的纯文件很大，使用散列版本会充分提高查找速度。

信息源按从左到右的顺序进行检查，直到其中一个给出了查询的答案。在上面的例子中，**gethostbyname** 例程首先检查**/etc/hosts** 文件，如果那里没有列出主机，就再检查 DNS。另一方面，对 UNIX 组的查询，**gethostbyname** 命令只检查**/etc/group** 文件。

如果需要，可在某个信息源后的中括号内更明确地定义它“失败”后的动作。例如：

```
hosts:      dns  [NOTFOUND=return]  nis
```

如果能用 DNS，就只用 DNS，即如果名字服务器返回一条否定的响应，则查询立刻返回（带有失败代码），并不检查 NIS。但是，如果没有名字服务器可用，则查询 NIS。失败的各种类型如

[5] 原来 NIS 的一种不成功的后续，现在 Sun 已经停止开发，但有些系统出于历史原因仍然支持它。

www.TopSage.com

表 17.3 所示，每种类型都可设置成 return 或 continue，表明查询是放弃还是继续转给下一个信息源。

表 17.3 /etc/nsswitch.conf 认识的失败模式

条 件	含 义	条 件	含 义
UNAVAIL	信息源不存在或者已关闭	TRYAGAIN	信息源存在，但处于忙状态
NOTFOUND	信息源存在，但不能回答查询	SUCCESS	信息源能够回答查询

在默认情况下，Linux 的各个发行版本所带的 **nsswitch.conf** 文件对于没有 NIS 独立运行的机器来说都适用。所有的信息源都归结到纯文件上，例外的是主机查询，它是先查纯文件，再查 DNS。我们举例的发行版本都默认对 **passwd** 和 **group** 采用 compat 模式，改动它或许还是值得一试的。如果您确实在用 NIS，只要把它加入 **nsswitch.conf** 文件即可。



Debian 和它的“近亲”Ubuntu 都把 protocols、services、ethers、rpc 设为先查 db，后查 files。这稍有些奇怪，因为 Debian 和 Ubuntu 实际上没有**/var/db**，也没有任何维护它的机制。直接到 files 大概效率会稍微高一些，如果愿意，您可以修改设置这么做。

17.4.4 网络组

NIS 引入了一个抽象的概念，即网络组（netgroup）。网络组是为了便于在其他系统文件中容易引用，而给用户、机器和网络起的组名。它们在文件**/etc/netgroup** 中定义，也作为一个 NIS 映射文件进行共享。

netgroup 项的格式为：

```
groupname  list-of-members
```

各成员之间用空白隔开。一个成员要么是一个网络组名，要么是以下形式的三元组：

```
(hostname,  username,  nisdomainname)
```

三元组中的任何空字段都是一个通配符，如（boulder,,）是指主机 boulder 上所有域中的所有用户（或者主机 boulder 本身，这取决于使用网络组名的环境）。字段中的短线表示否定，如(boulder,-,) 是指机器 boulder，且没有用户。网络组定义可以嵌套。

下面给出**/etc/netgroup** 文件的一个简单例子：

```
bobcats       (snake,,) (headrest,,)
servers       (anchor,,) (moet,,) (piper,,) (kirk,,)
anchorclients (xx,,) (watneys,,) (molson,,)
beers         (anchor,,) (anchor-gateway,,) anchorclients
allhosts      beers bobcats servers
```

这些网络组均用主机名来定义，这在实际中是典型的方式。

网络组可用在几个定义访问权限的系统文件中。当前最常见的应用是配置 NFS 的导出。在文件**/etc/exports** 中，可以用网络组来指定允许安装每个文件系统的主机组。当您向许多主机导出文件系统时，特别是在那些要求提供完整域名而又限制了 **exports** 文件每行不超过 1024 个字母的系统上，这一特性很方便。参考第 16 章了解更多有关 NFS 的信息。

网络组是一个很好的主意。它们能简化系统文件，使其更易于理解。它们还引入了一种间接方法，能在 1 个而非 15 个文件中修改用户或机器状态。

www.TopSage.com

17.4.5 建立 NIS 域

NIS 必须在主服务器、从服务器以及每台客户机上进行初始化。完成初始化需要两步过程。首先，在每台服务器都运行 **ypinit**。其次，域中的每台机器都要从一个系统启动文件中设置域名，并且配置文件**/etc/nsswitch.conf** 导入 NIS 数据。

一、配置 NIS 服务器

NIS 的服务器端通常必须以一个独立的可选软件包 **ypserv** 来安装。Debian 和 Ubuntu 有点儿不一样，它们的软件包 **nis** 中既包括客户器端也包括服务器端。

ypinit 初始化域中的主、从服务器。主服务器使用下列命令初始化：

```
# cd /var/yp              /* The NIS directory, wherever it is */
# domainname foo          /* Name the new domain. */
# /usr/lib/yp/ypinit -m   /* Initialize as master server. */
# ypserv                  /* Start the NIS server. */
```

-m 标记告诉 **ypinit** 正在配置主服务器，它将提示输入从服务器的一个列表。一旦主服务器打开并运行，就应该运行带有**-s**（代表 slave）标志的 **ypinit**，配置每台从服务器。

```
# cd /var/yp
# /usr/lib/yp/ypinit -s master /* Argument is master's hostname. */
# ypserv
```

ypinit -s 在本地复制一份主服务器当前数据的副本，出现的域数据文件足以让 **ypserv** 知道它应该服务该域。

在每台从服务器上，都应该建立 crontab 项，从主服务器拉出所有映射文件的新副本。**ypxfr** *map* 命令（其中 *map* 是名字，如 **passwd.byuid**）将从主服务器传输指定的映射文件。必须为每个映射文件运行一次该命令。由于映射文件修改的速率不同，如果网络带宽紧张，您可能想让某些映射文件传输次数多些，而另一些少些。大多数情况下，一天传输所有映射文件 1~2 次（可能是在晚上）便已足够。下面的脚本传输每个映射文件：

```
#!/bin/sh
mydomain = `/bin/domainname`
cd /var/yp/$mydomain   # the NIS directory
for map in `/bin/ls`; do
    /usr/lib/yp/ypxfr $map
done
```

在**/usr/lib/yp** 下还有预先写好的脚本（**ypxfr_1perday**、**ypxfr_2perday** 和 **ypxfr_1perhour**），以各种不同的频度传送 NIS 映射文件。

如果想让用户能利用 **yppasswd** 改变他们的口令，必须在主 NIS 服务器上运行 **yppasswdd** 守护进程。人们已经知道这个服务器的 Linux 版本会频繁崩溃，所以如果 **yppasswd** 显得似乎不干活了，那么就要核实它是否还在运行。

二、配置/etc/ypserv.conf 中的访问控制选项

您可以在**/etc/ypserv.conf** 中设置守护进程 **ypserv** 在 Linux 上的版本，不过只定义了几个选项，大多数网点都不需要改变它们的默认值。

更为重要的是，**ypserv** 查看 **ypserv.conf** 这个文件获得指示，知道如何控制对 NIS 数据的访问。Linux 的 **ypserv** 和传统的实现不一样，它不是简单地应答每个发来的查询请求，而是要对照访问列表检查发来的请求。每一个控制行的形式如下：

www.TopSage.com

```
host:nisdomain:map:security
```

host、*nisdomain* 和 *map* 区分出请求的一个特殊子集，而参数 *security* 表明如何处理它：deny 表示拒绝请求；port 表示只要请求是从某个特权网络端口（<1024）发来的就应答；而 none 表示应答请求。下面是一个配置的例子：

```
128.138.24.0/255.255.252.0:atrustnis:*:none
*:*:passwd.byuid:deny
*:*:passwd.byname:deny
128.138.:atrustnis:*:port
*:*:*:deny
```

您可以在 *host*、*nisdomain* 和 *map* 字段里用一个星号来匹配任意值，但是不允许做部分匹配（例如，您不能使用 passwd.*来匹配所有**/etc/passwd** 文件里的映射）。控制行要按顺序检查，直到找到一个匹配行。如果没有找到匹配行，那么默认会响应请求。

参数 *host* 可以包含一个网络掩码，就像上面例子中的第一行那样，但是 **ypserv** 不能理解更常用的 CIDR 记法。正像第 4 行那样，您也可以省略 IP 地址的后面部分，让 **ypserv** 用 0 填充，并提供一个类似的网络掩码。参见 12.4.4 节对 CIDR 掩码的讨论。

上例中的规则允许从 128.138.24/22 这个网络中的任何一台主机进行访问。只要请求是从一个特权端口来的，那么在 128.138 这个网络中的主机就能够访问 atrustnis 中的所有映射，但不包括/etc/passwd 文件中获得的映射。其他所有访问都被拒绝。

绝不要忘记一点，这种类型的访问控制至多算一种变通方法。它可能会引诱外单位的人偶尔浏览信息，但是它没有提供一种非常有效的办法抵御刻意的攻击者。

处于历史原因，这里也支持一种比较老的安全机制，即**/var/yp/securenets** 文件。新的配置应该使用 **ypserv.conf** 文件。

三、配置 NIS 客户机

设置完服务器后，必须通知每台客户机它已经是新域的一个成员了。域中的服务器通常也是客户机。

domainname 命令设置机器的 NIS 域。它在机器引导时由一个启动脚本运行。配置它所需的确切调整随发行版本不同而异，特定的发行版本的细节在以后介绍。参考第 2 章了解有关系统启动脚本的更多信息。

每个客户机都至少有 **passwd**、**group** 和 **hosts** 文件的一个最小私用版本。需要 **passwd** 和 **group** 允许 root 在没有 NIS 服务器可用时能登录。它们应该包含标准的系统账号和组：root、bin、daemon、wheel 等。在 NIS 启动并运行前，必须要有 **hosts** 文件（或者 DNS）能回答发生在引导时的查询。

四、各个发行版本的 NIS 细节



在 Fedora 和 RHEL 下，您可以通过设置**/etc/sysconfig/network** 中的变量 NISDOMAIN 来设置 NIS 域名。NIS 的服务器端用一个独立的软件包 **ypserv** 安装。**chkchonfig** 命令可以启用和禁用 **ypbind**、**ypserv** 和 **yppasswdd**，例如：

```
# chkconfig ypbind on
```



SUSE 在系统引导时从**/etc/domainname** 这个文件设置 NIS 域名。NIS 的服务器端用一个独立的软件包 **ypserv** 安装。**chkconfig** 可以让系统在引导时自动启动 **ypserv** 或 **ypbind**。您可以在**/etc/sysconfig/ypbind** 中设置 **ypbind** 的命令行选项。您必须将这个文件中的 YPBIND_BROADCAST 设置为 NIS 域名，或者安装一个**/etc/yp.conf** 文件，否则，启动脚本将会拒绝启动 **ypbind**。

www.TopSage.com



Debian 和 Ubuntu 把 NIS 域的名字保存在**/etc/defaultdowmain** 里。如果有**/etc/defaultdomain** 这个文件，那么系统的启动脚本会自动运行 **ypbind**。要运行 **ypserv**，可以编辑文件 **/etc/init.d/nis**，把 `NISSERVER` 的值设为 `slave` 或者 `master`。

17.5 LDAP：轻量级目录访问协议

UNIX 和 Linux 网站需要采用一种较好的途径分发管理性的配置数据，然而，问题实际上远不止于此。像电话和电子邮件通信录这样的非管理性数据怎么分发？您想与外界共享的那些信息怎么分发？每个人实际需要的是一种通用性的目录服务。

目录服务就是满足一些前提的数据库。与前提相吻合的任何数据集都有可能包括在数据库中。基本前提有下列几条：

- 数据对象相对较小；
- 数据库可被广泛复制与缓存；
- 信息是基于属性的；
- 频繁读取数据，但很少写数据；
- 搜索是一种共同的操作。

目前符合以上条件的 IETF 标准系统是 LDAP（Lightweight Directory Access Protocol，轻量级目录访问协议）。LDAP 规范实际上并不是指数据库本身，而是规定如何通过网络访问数据库。但因为它们规定了如何系统化数据以及如何执行搜索操作，所以也就暗示了一种相当特殊的数据模型。

LDAP 最初的设计是作为一种简单的网关协议，它能让 TCP/IP 客户机与 OSI 系统上运行的 X.500 目录服务器对话。随着时间的推移，不但 X.500 已经过时，而且 UNIX 迫切需要某种标准目录服务。这些因素导致 LDAP 朝着完善的目录系统的方向发展（这使它可能不再与 L 所代表的轻量级的含义相称）[6]。

目前（在 2006 年），LDAP 已经成为主流，或许部分是因为微软采用 LDAP 作为其 Active Directory 服务的基础。在 UNIX 和 Linux 这方面，OpenLDAP 软件包（www.openldap.org）已经成为标准实现。

17.5.1 LDAP 数据的结构

LDAP 数据采用属性列表的形式，用 LDAP 的术语叫“entry（条目）”。每个条目包括一组有名字的属性（比如“uid”或者“description”）以及这些属性的值。Windows 用户可能觉得这种结构和 Windows 的注册表很像。和注册表里的情形一样，一个属性可以有多个值。

下面是一个例子，它是用一个 LDAP 条目来表示**/etc/passwd** 文件一行内容的典型（但经简化的）形式：

```
uid: ghopper
cn: Grace Hopper
userPassword: {crypt}$1$pZaGA2RL$MPDJoc0afuhHY6yk8HQFp0
loginShell: /bin/bash
uidNumber: 1202
gidNumber: 1202
homeDirectory: /home/ghopper
```

[6] 因为 LDAP 曲折的历史，许多材料都倾向于详细解释 LDAP 的 X.500 和 OSI 的联系。不过，这段历史和 LDAP 当前的用途没什么关系。所以可忽略不计。

www.TopSage.com

这种表示方式是 LDIF 的一个简单例子，LDIF 是指 LDAP 数据交换格式（LDAP Data Interchange Format），大多数与 LDAP 有关的工具和服务器实现都采用了这种格式。LDAP 数据很容易从纯文本来回转换，这个事实也是 LDAP 获得成功的部分原因。

LDAP 条目通过使用“专有名称”（属性名：dn）组织成一种层次结构，dn 构成了一种搜索路径。例如，上述用户的 dn 可以是：

```
dn: uid=ghopper,ou=People,dc=navy,dc=mil
```

和 DNS 中的情况一样，“最高有效位”在右边。在这里，一直用 DNS 名 navy.mil 做 LDAP 层次结构的最高级。这个 DNS 名分成两个域名成分（dc），即“navy”和“mil”，但这只是几种常见约定之一。

每个条目只有一个专有名。因此，条目的层次结构看上去就像一个简单的无环有叉树。不过，有办法在条目之间建立符号链接，也有办法指向其他服务器。

LDAP 条目一般通过使用一个 objectClass 属性来表示。对象类指出了一个条目能够包含的属性，其中有些属性必须要有。这种方式也给每个属性分配了一个数据类型。对象类能以传统的面向对象的方式进行嵌套和组合。对象类树的最高层类叫做“top”，它仅指出一个条目必须有一个 objectClass 属性。

表 17.4 列出了一些常见的 LDAP 属性，这些属性的含义并不是很直观。

表 17.4 LDAP 层次结构里的一些常见属性

属 性	代 表	含 义
o	机构（Organization）	经常标识一个站点的顶级条目
ou	机构单元（Organizational unit）	逻辑上的组成部分，例如“marketing”
cn	常用名（Common name）	表示条目的最自然的名字
dc	域名成分（domain component）	用在以 DNS 为模型建立 LDAP 层次结构的站点
objectClass	对象类（Object Class）	该条目的属性所遵循的模式

17.5.2 LDAP 的特点

在您对 LDAP 已经有了一些经验之前，它都是一个难以掌握的棘手概念。和 NIS 不同，LDAP 本身没有解决任何特定的管理问题。LDAP 没有专门适用的“主要任务”，各个站点部署 LDAP 服务器的理由也五花八门。所以，在我们谈到安装和配置 OpenLDAP 的特定之处以前，或许值得回顾一下研究在您的站点使用 LDAP 的一些原因。下面是几个主要原因。

- 您可以使用 LDAP 替换 NIS。它比 NIS 好的地方很多，但是主要优点是安全更好、工具支持得更好，而且灵活性更强。
- 您也能以类似方式，将 LDAP 用于分发传统的 NIS 域以外的配置信息。包括 sendmail 和 Postfix 在内的大多数邮件系统，都能从 LDAP 获得大部分寻径信息，这实际上也是 LDAP 最流行的应用领域之一。从 Apache Web 服务器到 autofs 的自动安装器这样的多种工具也都能配置成使用 LDAP。随着时间推移，LDAP 可能会变得越来越常见。参见 18.5.2 节了解有关 sendmail 如何使用 LDAP 的知识。
- 用 **ldapsearch** 这样的命令行工具可以方便地访问 LDAP 数据。此外，常见的脚本语言，如 Perl 和 Python（通过使用库）都能很好地支持 LDAP。因此，对于本地编写脚本和管理性工具来说，LDAP 是分发配置信息的一种很好方式。
- LDAP 有很出色的基于 Web 的管理工具，例如 phpLDAPadmin（phpldapadmin.sourceforge.net）和 Directory Administrator。这些工具都易于使用，只要安装好，不用看手册页就能开始用。
- 作为一种公开的目录服务，LDAP 得到了很好的支持。绝大多数主要的电子邮件客户端软件，

www.TopSage.com

包括在 PC 上运行的那些软件，都支持用 LDAP 访问用户的目录信息。许多 Web 浏览器通过使用一种 LDAP 的 URL 类型，也能支持做简单的 LDAP 搜索。

- 微软的 Active Directory 基于 LDAP，目前发布的 Windows 2003 Server R2 所包括的扩展（原来叫做“Services for UNIX”，现在叫做“Windows Security and Directory Services for UNIX”）能很方便地映射 Linux 的用户和用户组。

17.5.3 LDAP 的文档和规范

我们还不知道有哪一种资料确实能很好地介绍 LDAP 总体的体系结构。一次性的 FAQ 也不再得到维护。目前最好的入手资料是 Open LDAP 软件包的配套文档和 Linux 的 *LDAP Howto*。本章末尾还举出了两本比较好的 LDAP 书籍。

与 LDAP 相关的 RFC 数量大而且种类多。这些 RFC 构成的一组，表现出极大的复杂性，而且用途有点儿没有代表性。表 17.5 中列出了一些重要的版本。

表 17.5 与 LDAP 有关的重要 RFC

RFC	标 题
4519	LDAP: Schema for User Applications
4517	LDAP: Syntaxes and Matching Rules
4516	LDAP: Uniform Resource Locator
4515	LDAP: String Representation of Search Filters
4514	LDAP: String Representation of Distinguished Names
4513	LDAP: Authentication Methods and Security Mechanisms
4512	LDAP: Directory Information Models
4511	LDAP: The Protocol
3672	Subentries in the Lightweight Directory Access Protocol (LDAP)
3112	LDAP Authentication Password Schema
2849	LDAP Data Interchange Format (LDIF)—Technical Specification
2820	Access Control Requirements for LDAP
2307	An Approach for Using LDAP as a Network Information Service

17.5.4 OpenLDAP：Linux 的 LDAP

OpenLDAP 是对原来密执安大学（University of Michigan）所做工作的一种扩展，现在继续作为一个开放源代码项目存在。大多数 Linux 发行版本都带有 OpenLDAP，尽管默认安装并不要求包括它。OpenLDAP 的文档或许可以用“爽”字来形容。

在 OpenLDAP 的软件发布中，**slapd** 是标准的服务器守护进程，而 **slarpd** 处理复制。有一组选出的命令行工具能够查询和修改 LDAP 数据。

OpenLDAP 的配置很直观。首先，通过复制一份随 OpenLDAP 服务器一起安装的配置文件样本，就可以创建**/etc/openldap/slpad.conf** 文件。需要注意下面这几行：

```
database bdb
suffix "dc=mydomain, dc=com"
rootdn "cn=admin, dc=mydomain, dc=com"
rootpw {crypt}abJnggxhB/yWI
directory /var/lib/ldap
```

www.TopSage.com

数据库格式默认采用 Berkeley DB，它适合于 OpenLDAP 系统里存在的数据。您可以使用各种各样的后端，包括随时创建数据的脚本这样的特别方法。

`suffix` 是“LDAP 基准名”。它是 LDAP 名字空间在这里的根，在概念上类似于 DNS 域名。这个例子演示了一种常见的做法：用您的 DNS 域名作为您的 LDAP 基准名。

`rootdn` 是系统管理员的名字，`rootpw` 是系统管理员采用 UNIX 格式（DES）的口令。注意，还必须一直指定到系统管理员名字的域的各个成份。您可以用从**/etc/passwd** 文件“复制和粘贴”这个口令（如果没有使用 MD5 口令的话），也可以用一个只有一行的简单 Perl 脚本生成它：

```
perl -e "print crypt('password', 'salt');"
```

这里的 *password* 是所需要的口令，*salt* 是一个任意的双字符的字符串。因为出现了这个口令，所以要确保 **slapd.conf** 文件的权限是 600，它属于 root 所有。

您还需要编辑**/etc/openldap/ldap.conf** 文件，设置 LDAP 客户机请求的默认服务器和基准名。这做起来相当直接——只要把 `host` 条目的参数设置为服务器，而将 `base` 设置为等于 **slapd.conf** 中 `suffix` 的值就可以了（注意这两行都没有被注释掉）。

此刻，不带参数运行 **slapd**，应该就可以启动它了。

17.5.5 用 LDAP 代替 NIS

实际上不需要用 NIS“迁移”到 LDAP，但是因为 NIS 定义了共享系统文件的一个标准集合，它能作为一个有用的参照点。参见 20.5.6 节了解更多有关 PAM 的知识。

能够通过 NIS 分发的文件都已经由名字服务切换文件（**/etc/nsswitch.conf** 文件）“虚拟化”了，所以客户端对 LDAP 的支持相当容易加。有些 Linux 发行版本默认安装了必要的 **nss_ldap** 软件包，该软件包通常作为备选软件提供。这个软件包包含一个 PAM 模块，除了名字服务切换文件，还能让您使用带有 PAM 的 LDAP。

这个软件包的 LDAP 客户端默认值在**/etc/ldap.conf** 里设置，它和上一节介绍的**/etc/openldap/ldap.conf** 文件格式相同，但是它包含专门针对名字服务和 PAM 环境的其他选项。您还必须要在每个客户机上编辑**/etc/nsswitch.conf** 文件，加上 `ldap`，作为每种您想要 LDAP 化的数据来源（**nsswitch.conf** 文件的变化让 C 语言库把请求传给 **libnss_ldap** 库，后者接着用**/etc/ldap.conf** 里的信息推断出如何执行 LDAP 查询）。

RFC2307 规定了从传统的 UNIX 数据集（如 **passwd** 和 **group** 文件）到 LDAP 名字空间的标准映射。对于用 LDAP 代替 NIS 的系统管理员来说，这是一份至少从理论上讲很有用的参考文献。实际中，计算机读这份规范比人要容易得多，您最好看看例子。

Padl Software 提供了一套免费的 Perl 脚本，它们将现有的纯文件或者 NIS 映射文件转为 LDAP。从 www.padl.com/tools.html 可以取得它们，执行这些脚本很方便。可以把它们用作过滤器来生成 LDIF，还可以对一个活动的服务器执行它们，直接上载数据。例如，脚本 **migrate_group** 把**/etc/group** 中的这行：

```
csstaff:x:2033:evi,matthew,trent
```

转为下面的 LDIF：

```
dn: cn=csstaff,ou=Group,dc=domainname,dc=com
cn: csstaff
objectClass: posixGroup
objectClass: top
userPassword: {crypt}x
gidNumber: 2033
memberuid: evi
memberuid: matthew
memberuid: trent
```

www.TopSage.com

（注意，对象类和专用名称的规范，17.5.1 节中举例的 **passwd** 省略了它们。）

导入数据库之后，可以运行 **slapcat** 工具显示完整的数据库，以此验证传送正确。

17.5.6 LDAP 和安全

以前，LDAP 更多的是用作一个电话通信录，而没有用在别处，出于这个目的，数据不经加密就发送也往往是可以接受的。因此，“标准”的 LDAP 实现允许通过 TCP 端口 389 不加密就能访问。不过，我们强烈建议不要用没有加密的 LDAP 传输身份验证信息，即使口令单独用散列或者加密处理过也不行。

还有另一种方法，在大多数情况下（包括在微软的环境里）都能在客户机和服务器两端用 LDAP-over-SSL（称为 LDAPS，通常运行在 TCP 端口 686 上）。这种访问方法更可取，因为它保护了查询和响应中所带的信息。尽可能使用 LDAPS。

一个有着和 LDAP 一样的复杂性和一样多的不定组成部分的系统，不可避免地会有大量的意外误配的可能性，削弱系统的安全性。当然，也还可能会有一些原来明显的安全漏洞。系统管理员要小心。

17.6 推荐读物

MALÈRE, LUIZ ERNESTO PINHEIRO. *LDAP Linux HOWTO.*

VOGLMAIER, REINHARD. *The ABCs of LDAP: How to Install, Run, and Administer LDAP Services.* Boca Raton, FL: Auerbach Publications, 2004.

CARTER, GERALD. *LDAP System Administration.* Sebastopol, CA: O'Reilly Media, 2003.

17.7 习题

E17.1 为什么用拉的方法更新本地机器的文件比推的方法要更安全？

E17.2 解释下面取自一个 **rdist** distfile 文件的一段内容：

```
LINUX_PASSWD = ( redhatbox debianbox susebox )

passwd:
    ( /etc/passwd ) -> ( ${LINUX_PASSWD} )
    install /etc/passwd.rdist;
    cmdspecial /etc/passwd.rdist "/usr/local/sbin/mkpasswd";
```

★E17.3 说明 **rdist** 和 **rsync** 之间的不同。在哪些场合下用其中一种比另一种更好？

★E17.4 比较 NIS 和 LDAP。在什么时候您会用其中一种而不用另一种？您会两种都用吗？

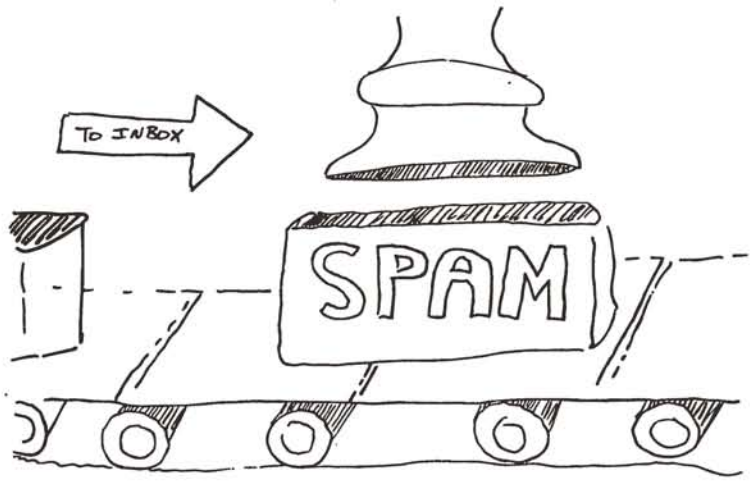
★E17.5 您的网点采用了什么方法来共享系统文件？和该方法相关的安全问题是什么？为您的网点提出另一种共享系统文件的方法，详细说明它所解决的问题。有什么缺点吗？

★★★★★E17.6 设计一种 LDAP 方案，保存诸如登录、口令、shell、授权主机等用户信息。构建一个工具，以交互方式把新用户输入到数据库中，或者从一个包含有用户清单的文件里输入数据库。构建一个工具，从 LDAP 数据库为实验室内的计算机产生 **passwd**、**group** 和 **shadow** 文件。如果用户愿意，允许他们在每台机器上有不同的口令（不是所有的用户都必须得到授权才能使用每台计算机）。您的 **adduser** 系统应该能打印出现有用户的登录名，还能打印出新用户的登录名和口令。

www.TopSage.com

大家网
TopSage.com

第 18 章　电子邮件



很难记起没有电子邮件的世界是个什么样子了。从学校里的孩子到老奶奶，再到公司里最落伍的员工，每个人现在都会例行公事般地使用电子邮件联系家人、同事、伙伴、客户甚至政府。真是一个电子邮件的疯狂世界[1]。

电子邮件之所以流行，是因为公众能够轻而易举地理解发一则消息的概念，这个概念的模型与传统的书面信件非常近似。电子邮件就“那么简单”，如果您知道某个人的电子邮件地址，您就可以敲一则消息，写上他们的地址，然后按“发送”。哇！几秒之后，这则消息就送到了他们的电子邮箱里，不管他们就在隔壁还是半个地球以外。从用户的角度来看，没有比这更容易的了。

遗憾的是，事情不总是这样的容易，即使在今天，保证在这样大的范围内实现电子邮件通信所依靠的下层基础设施的任务仍然相当繁重。您可以在 Linux 上运行几种软件包来传输和管理电子邮件（本章后面讨论其中的 3 种），但是它们都要求做一定程度的配置和管理。此外还有一点很重要，就是您要理解与电子邮件相关的底层概念和协议，从而不会给自己的用户养成错觉，以为在单位之间跨平台的电子邮件是老天送来的礼物，每次都能奇迹般地做到。

现如今，理解和管理电子邮件基础设施还有另外一些方式。现在一些服务提供商提供了“被管”电子邮件服务，电子邮件实际上托管在很远之外的数据中心里提供商的服务器上，您按月或者按年（可能还要按人）支付访问费用。类似地，还有一些免费的“托管”服务，如 Yahoo！Mail、MSN Hotmail 和 Google 的 Gmail 都已

[1] 甚至 Evi（本书作者之一）畅游在偏远的海上时，她也几乎一直能通过她的 BlackBerry 网卡保持电子邮件联络，这要感谢无处不在的 GPRS 网络。GPRS 网络遍布 113 个国家，但不算佛蒙特州！（译者注：在作者撰写本书的时候，佛蒙特州的“脱离美国”运动正闹得沸沸扬扬。）

www.TopSage.com

经在个人中非常流行。如果您正在为个人找一个电子邮件账号，或者在为一家（非常）小的公司找一个账号，那么这些都可供您选择。这些服务卸掉了若干负担，举例来说包括存储、服务器管理、软件升级、配置、垃圾邮件过滤和保持安全警惕。作为对其“免费”服务的回报，或许您要看到一些广告。这似乎在许多情况下都很理想，如果这样的选择适合于您，那么您至少能得益于不用读本章余下的内容啦。

不过，托管的电子邮件并不是每个人的解决方案。商业公司，以及其他依靠电子邮件服务来运行的大型机构，往往都不会冒风险去把电子邮件托管到异地。这样的机构可能有各种不同的理由运行自己的电子邮件系统，包括安全、性能和可用性。本章的内容就是面向这些人的。

本章篇幅较长，这也印证了电子邮件系统的复杂性。本章既包含了背景知识，也包含了软件配置的细节信息，讲述的顺序也大致如此。

我们曾试着把这一章分成较短的几章（邮件系统、**sendmail** 配置、垃圾邮件、Exim 和 Postfix），但这样写会在内容上充满了死循环问题，令人不好理解，而且我们以为，这样做用处不大。我们代之以在表 18.1 中给出带注释的目录。

表 18.1　　本章的编排

	小　节	内　容
背景知识	1	邮件系统及其各个组成部分
	2	寻址、地址语法、邮件信头
	3	原理、客户机/服务器的设计、邮件之家
	4	别名、邮件路由、邮递列表软件、LDAP
	5	邮递列表软件
sendmail 的配置	6	**sendmail**：安装、启动、邮件队列
	7	配置 **sendmail** 初探，**m4** 宏
	8	基本的 **sendmail** 配置原语
	9	高级的 **sendmail** 配置原语
	10	垃圾邮件，**sendmail** 访问数据库
	11	配置案例学习
	12	安全
	13	性能
	14	收集统计信息、测试和调试
其他	15	Exim，**sendmail** 的替代软件
	16	Postfix，**sendmail** 的另一种替代软件
	17	更多的资料来源

在依次阅读本章的内容时，这样的组织会使读者觉得内容更顺畅一些，但有时会被某些内容给打散了，这些内容介绍了某种与电子邮件相关的特定任务。对于中等规模单位的邮件服务负责人来说，可能需要阅读整章内容。如果只是为一般的公司客户设置 PC 支持电子邮件，对于这样的系统管理员来说，就没必要这么做了。

表 18.2 为一些常见的系统管理工作提供了一份引导指南。

表 18.2　　工作索引

工　作	小　节	工　作	小　节
升级 sendmail	6，7	为一个站点设计邮件系统	3，4，5，6，7，8，9，10，11
第一次配置 sendmail	3，6，7，8，9，10，11，12	反垃圾邮件	10
改变配置文件	7	审计安全性	12

www.TopSage.com

续表

工 作	小 节	工 作	小 节
设置PC收邮件	1，3	虚拟主机	9
设置邮递列表	5	使用Exim代替sendmail	15
性能调优	3，9，13	使用Postfix代替sendmail	16

本章的大部分内容都是在讲述**sendmail**的配置，它是分析和传递电子邮件的标准程序。**sendmail**最初是加州大学伯克利分校（University of California，Berkeley）的Eric Allman编写的。**sendmail**有3个主要的版本：版本5、IDA以及版本8。最近，全新设计的版本sendmail X已经发布了一个早期的beta版，但还没有准备好可以供生产环境使用（根据圈内人透露，它可能永远也代替不了sendmail 8）。版本5和IDA已经不常使用，而由版本8取代了它们。在本章中我们讨论版本8（精确地说是8.13）。

sendmail正在由Sendmail,Inc公司进行商业开发，该公司还维护着一个免费的、开放源代码的版本。商业版本的特色是图形化的用户配置界面以及中央监控和报告功能（这是对于用户数量庞大的邮件站点来说特别有用的功能）。

18.1 邮件系统

从理论上说，一个邮件系统由4个不同的部分组成。

- 让用户阅读和撰写邮件的“邮件用户代理”（mail user agent，MUA）。
- 在机器之间发送消息的“邮件传输代理”（mail transport agent，MTA）。
- 把消息放到本地消息库[2]中的“投递（delivery）代理”；它有时叫做本地投递代理（Local Delivery Agent，LDA）。
- 可有可无的“访问代理（access agent）”，它可以把用户代理连接到消息库（例如，通过IMAP或POP协议）。

有些站点还使用一种邮件提交（submission）代理，这种代理使用SMTP（simple mail transport protocol，简单邮件传输协议），并完成传输代理的一些工作。图18.1显示了这些组成部分的相互关系。

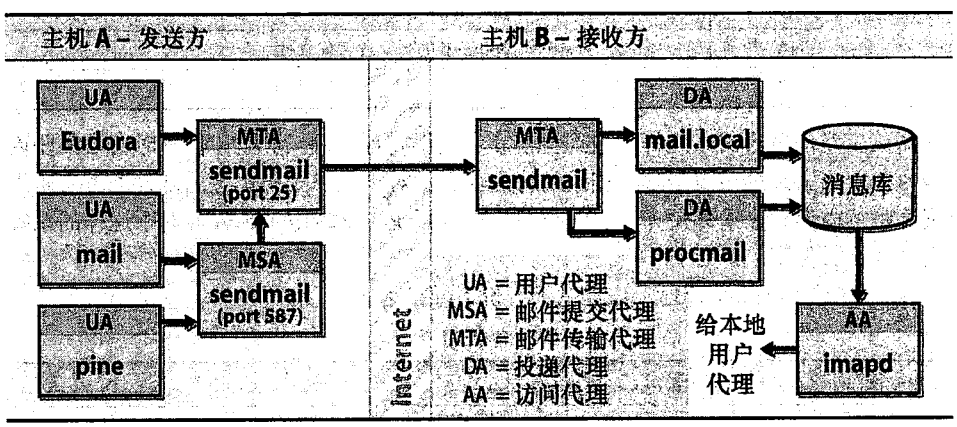


图18.1 邮件系统的组成部分

[2] 收件方用户的邮箱，或者有时是个数据库。

www.TopSage.com

18.1.1 用户代理

电子邮件用户使用用户代理来阅读和撰写消息。电子邮件消息最初只由文本组成，但现在用一种称为 MIME（Multipurpose Internet Mail Extensions，多用途网际邮件扩充协议）的标准可以把文字格式和附件（包括许多病毒）编码后加入到电子邮件中。大多数用户代理都支持它。因为它并不影响邮件的寻址或传送，所以我们不在本章中对它作进一步讨论。

用户代理的一项工作是确保在邮件消息的内容中嵌入的任何可能被邮件系统误解的文字得到保护。用作消息之间记录分隔符的字符串“From”就是一个例子。

/bin/mail 是最初的用户代理，对于从 shell 提示符下阅读文本类的电子邮件消息来说，仍然是个随时能用的好工具（译者注：原文 ol'standby 的意思是 online standby，即“在线待命”。自从知名网站 AOL 诞生之后，OL 在网上就成了 ONLINE 的流行缩写）。不论这是好是坏，Internet 上电子邮件已经超越了文本时代，所以对大多数用户来说，基于文本的用户代理不再实用了。图形用户界面支持以“鼠标指点”方式访问邮件消息，并且能很好地处理诸如图片、微软 Word 文档和电子表格之类的附件。

图 18.1 显示出的优秀特性之一就是，一个用户代理不必和邮件系统的其余部分运行在相同的系统，乃至相同的平台上。用户可以在登录到 Linux 台式机的时候，运行 Linux 带的许多用户代理中的一种，但他们也可以通过访问代理（AA）协议（例如 IMAP 或者 POP），从自己的 Windows 笔记本电脑访问他们的电子邮件。迄今为止，这是最常见的配置方式。为什么说 Windows 和 Linux 不能和睦共处呢？

下面列出了常用的用户代理及其最初的来源。

- Red Hat 和 Fedora 上的**/bin/mail** 是原 UNIX 命令 **mail** 的 BSD 版本；在 SUSE、Debian 和 Ubuntu 上，它变成了**/usr/bin/mail**[3]。这个用户代理只支持文本，并且需要在本地保存邮件。
- Mozilla 的 Thunderbird，有 Linux、Windows 和 Mac OS 的版本。
- Evolution（也叫做 Novell Evolution，以前叫 Ximian Evolution），有 Linux、Windows 和 Mac OS 的版本。
- 华盛顿大学（University of Washington）的 **pine**，www.washington.edu/pine。
- Qualcomm 的 Eudora，用于 Mac 机或者运行 Windows 的 PC。
- 微软的 Outlook，也用于 Windows。

18.1.2 传输代理

传输代理必须接受从用户代理那里来的邮件，读懂收件人的地址，并设法把邮件交给正确的主机进行投递。大多数传输代理还担当消息提交代理的角色，完成把新消息发送到邮件系统的功能。传输代理使用 RFC821 中定义的 SMTP 协议（Simple Mail Transport Protocol，简单邮件传输协议），或者使用 RFC1869、RFC1870、RFC1891 和 RFC1985 中定义的 ESMTP 协议（Extended SMTP，扩展的 SMTP 协议）。

UNIX 和 Linux 系统有好几种传输代理（主要有 PMDF、Postfix、**smail**、**qmail**、Exim 和 **zmailer**），但是 **sendmail** 是最全面、最灵活和使用范围最广的传输代理。2001 年进行的一项有关邮件系统的调查[4]显示，**sendmail** 占据了 60%的份额，Exim 为 8%，Microsoft Exchange Server 为 4%，以及 Postfix 为 2%，其他（它们大约有 50 种）则占据剩下的份额。

Red Hat、Fedora 和 SUSE Linux 自带装好的 **sendmail**。Debian 表面上好像是带了 **sendmail**，但是如果您仔细一看，会发现 **sendmail** 实际上是到 Exim 这个邮件传输代理的链接。Exim 经过

[3] 这个用户代理在其他系统上有时候称为 **Mail** 或者 **mailx**。于是，Red Hat 提供的链接叫做 **Mail**，而 SUSE、Debian 和 Ubuntu 提供的链接分别叫做 **Mail** 和 **mailx**。

[4] Matrix.net 为 Sendmail, Inc.公司做的保密调查。

www.TopSage.com

仔细地调整开发，已经能够理解 **sendmail** 的命令行标志。直接调用“**sendmail**”的用户代理应该是不知情的。Ubuntu 默认带 Exim。

18.1.3 投递代理

投递代理从传输代理那里接受邮件，并把它真正投递给适当的本地收件人。邮件可以投递给某个人、一个邮递列表、一个文件、甚至投递给一个程序。

每种类型的收件人可能需要一个不同的代理。**/bin/mail** 是用于本地用户的投递代理。**/bin/sh** 是发给文件或程序的邮件最初的投递代理，投递到文件则在内部处理。**sendmail** 的新近版本附带了更安全的本地投递代理，名为 **mail.local** 和 **smrsh**（念做“smursh”）。www.procmail.org 的 **procmail** 也可以用作本地投递代理，参见 18.9.16 节。类似地，如果您运行 Cyrus **imapd** 作为 AA（访问代理），它还包括自己的本地投递代理。

18.1.4 消息库

消息库是本地计算机上保存电子邮件的地方。它过去常常是目录**/var/spool/mail** 或**/var/mail**，其中邮件被保存在以用户的登录名命名的文件中。这仍然是最常见的消息库，但拥有数千或数百万电子邮件客户的 ISP 们正在寻找用于消息库的其他技术（通常是数据库）。

在使用**/var/spool/mail** 或**/var/mail** 库的系统中，邮件目录是在操作系统的安装过程中创建的。除非您使用 **mail.local** 作为您的本地邮寄程序（在这种情况下，它的权限设置可能为 755），否则应该把它的权限设置为模式 775，属组为 mail[5]。在我们举例的 Linux 平台上稍微有一点儿不同。

```
Red Hat: drwxrwxr-x   2   root   mail    1024 Dec  5 11:16  /var/spool/mail
Fedora:  drwxrwxr-x   2   root   mail    4096 Mar 17 08:42  /var/spool/mail
SUSE:    drwxrwxrwt   2   root   root    4096 Aug  2 23:25  /var/spool/mail
Debian:  drwxrwsr-x   2   root   mail    4096 Aug  3 16:17  /var/mail
Ubuntu:  drwxrwsr-x   2   root   mail    4096 Jan 8 03:22   /var/mail
```

SUSE 的权限要宽松一点儿，但是在邮件缓存目录中的文件模式为 660，属组为 root。设置了粘附位的目录（在权限位中的 t）不允许用户彼此删除对方的文件，即便他们有这个目录的写权限也不行。不过，恶意的用户可以填满邮件缓存目录，把它当作一个乱写乱画的分区，或者创建另一个用户的邮箱。参考 5.5.3 节了解有关粘贴附件的更多信息。

18.1.5 访问代理

像 **imapd** 和 **spop** 这样的程序是 PC、Mac 或者 Linux 用户的访问代理，这些用户的邮件先投递到一台 Linux 服务器上，然后再分别用 IMAP（Internet Message Access Protocol，Internet 消息访问协议）或者 POP（Post Office Protocol，邮局协议）下载。IMAP 和 POP 在 18.3.3 节介绍。

18.1.6 邮件提交代理

邮件领域的另一个新产品是高容量站点所必需的，这就是邮件提交代理（mail submission agent）。在繁忙的主邮件枢纽上的传输代理要花费大量时间进行邮件消息的预处理：确保所有主机名是完整的，

[5] 要对文件执行一次非 root 的 **chown** 命令才能投递邮件系统也要求这个目录有组的写权。一般而言，执行非 root 的 **chown** 命令是个不好的想法。

www.TopSage.com

修改从已损坏的邮件用户代理（MUA）那里得到的信头、日志记录错误、重写信头等。RFC2476 引入的思路是：将邮件提交代理（MSA）从邮件传输代理（MTA）中分离出来，以便分担工作负荷并获得最佳性能。

这种想法就是使用 MSA，它在另一个端口上运行，作为处理由本地用户代理注入到邮件系统中的新消息的一种“招待员”。MSA 负责消息由传输代理发送之前必须完成的所有预备工作和错误检验。它有点像在 MUA 和 MTA 之间插入一个头脑清楚的检验员。

特别地，MSA 可确保所有的主机名都是完整的，它验证本地主机名在加上本地域部分之前是合法的。如果消息头部丢失或不一致，MSA 还会修复它们。通常，MSA 会添加一个 From 或 Date 头部，或调整 Message-Id 的头部。MSA 能做的最后一项工作是把发件人的地址从登录名改写为更可取的外部形式，比如 *First_Last*（译者注：这是按照西方人的习惯，“first name”也就是名在前，“last name”也就是姓在后）。

为了让这个方案生效，必须把用户代理配置成在端口 587 而不是端口 25 上连接 MSA，后者是用于邮件的传统端口。如果用户代理不能识别端口 587，您仍然可以在端口 25 上运行 MSA，但是得在和 MTA 不同的另一台服务器。您还必须配置 MTA，使得它不会重复 MSA 已完成的工作。重复处理不会影响邮件处理的正确性，但它确实意味着无用的额外工作。

在默认情况下，**sendmail** 既可以担当 MSA 的角色也可以担当 MTA 的角色。从 **sendmail** 8.10 开始，运行一个单独命令就能监听 25 端口和 587 端口。用户代理经常直接带标志来调用 **sendmail**，要求它接受一则邮件消息（**-bs** 或者**-bm**），或者根本就不带标志，在后一种情况下，**sendmail** 默认以**-bm** 标志运行。**sendmail** 进程跟踪调用它的方式，如果用**-bs** 或者**-bm** 标志调用它，它就变成 MSA，如果用**-bd** 标志调用它，它就变成 MTA。

直接打开 SMTP 连接的用户代理必须进行修改，使用 587 端口来调用 MSA。

18.2 剖析邮件消息

在深入研究 **sendmail** 配置之前，我们必须理解邮件消息的 3 个不同部分：

- 信封（envelope）；
- 信头（header）；
- 消息主体（body of message）。

信封确定消息将投递到哪里，或者如果消息不能投递出去的话，应该把它返回给谁。对于单个收件人来说，信封地址通常与信头的 From 和 To 行一致，如果消息是发送给一个邮递列表的，那么就不一致了。地址是单独提供给 MSA 的。信封对用户不可见，它不是消息本身的一部分，它供 sendmail 在内部确定把这则消息发送到哪里。

信头是一组格式遵循 RFC822 的属性/值。它们记录与消息有关的各种信息，比如发送的日期和时间、它在旅程中经过哪些传输代理传递等。信头是邮件消息的一个真实部分，只不过用户代理向用户显示消息时通常会隐藏一些不太引人注意的项。

消息的主体是将要发送的实际内容。它必须由纯文本组成，虽然文本经常表示各种二进制内容，但此时采用了对邮件来说是安全的编码方式。

当我们在介绍配置的小节时，有时会谈到信封发件人和收件人，有时则会谈到信头发件人和收件人。我们总是试图在上下文中不能确定的地方指出我们正在谈论的是哪种地址。

18.2.1 邮件寻址

本地寻址很简单，因为用户的登录名就是一个唯一的标识符。Internet 地址也很简单：

www.TopSage.com

user@host.domain 或者 *user@domain*。在电子邮件和 Internet 的早期时代，表 18.3 列出的地址形式是很常见的。

表 18.3 过时的地址类型举例

地址类型	地址的例子	现代的形式
UUCP	mcvax!uunet!ucbvax!hao!boulder!lair!evi	evi@lair
基于路径	<@site1,@site2,...,@siteN:user@final-site>	user@final.site
“百分号扩展”	user%host1%host2@host3	user@host1

sendmail 配置中的复杂性大多源自早期对处理这样的地址的要求。这些寻址形式都依赖于中继转发（relaying），而幸亏有了发垃圾邮件的家伙（spammer），各站点逐渐关闭了中转功能。spammers 企图隐藏自己的身份，或通过您的机器中转邮件，“百分号扩展（percent hack）”（表 18.3 中最后一行）是他们最喜欢的一种工具。如果您需要处理这里面的任何一种地址形式，请参考 **sendmail** 文档，寻求帮助。

18.2.2 阅读邮件信头

每一则邮件消息都从称为信头（header）的若干行开始，信头中包含着有关这则消息的信息。每个信头以一个关键字打头，比如 To、From 或 Subject，后面跟着一个冒号和该信头的内容。标准信头的格式在 RFC822 中定义，不过，自定义信头也是允许的。任何以“X-”打头的信头都将被邮件系统忽略，但会随着消息传送。因此，您可以给电子邮件消息添加一个信头，比如 X-Joke-of-the-Day，这不会干扰邮件系统发送它们的能力[6]。

有些信头是由用户代理添加的，另一些则是由传输代理添加的。有些信头会记录下消息通过邮件系统的路径。许多用户代理会对您隐藏这些“人们不感兴趣”的信头，但通常可以选择让代理把它们全部显示出来。当我们被垃圾邮件轰炸，而有时必须回溯一则消息的来源时，阅读信头就成为了一项重要的技巧。下面是来自一则简单消息的信头部分：

```
From trent Fri, 30 Jun 2006 20:44:49 -0600
Received: from bull.atrust.com (bull.atrust.com [127.0.0.1]) by bull.atrust.com
    (8.13.1/8.13.1) with ESMTP id k612inkG001576 for <ned@bull.atrust.com>;
    Fri, 30 Jun 2006 20:44:49 -0600
Date: Fri, 30 Jun 2006 20:44:48 -0600
From: trent@atrust.com
Message-Id: <200607010244.k612im9h001575@bull.atrust.com>
To: ned@bull.atrust.com
Cc: steve@bull.atrust.com
Subject: Yonder Mountain

------ body of the message was here ---
```

这则消息完全停留在本地计算机上，发件人是 trent，而收件人是 ned。第一个 From 行是由 **mail.local** 添加的，**mail.local** 此时是本地投递代理。信头中的 Subject 和 Cc 两行是由 trent 的邮件用户代理加的，它可能还会加上信头中的 To、From 和 Date 行。如果 MUA 没有提供信头中的 To、From 和 Date 行的话，那么邮件传输代理 **sendmail** 会加上它们。每台接触到这则消息的机器（或者更精确地说，是每个 MTA）都会添加一个 Received 信头。

邮件消息中的信头包含着有关这则消息到过哪里，它在那里停留了多长时间，以及它最终是何时投递到其目的地等大量信息。下面是通过 Internet 发送的一则邮件消息的更完整的剖析。其中穿插了

[6] 从技术的角度讲，您可以随意增加任何信头，因为路由邮件只使用信封，而忽略信头。

www.TopSage.com

注释，用来描述各种信头的用途并标识出添加它们的程序。左边的行号是为了方便在下面的讨论中引用，而并非消息的一部分。某些行折成了好几行，这是为了让例子能在一页里显示得下。

```
1: From eric@knecht.sendmail.org
```

第 1 行由**/bin/mail** 或 **mail.local** 在最后一次投递的过程中添加，用于将这则消息与收件人邮箱中的其他消息区分开。有些邮件阅读器是通过寻找后面跟着字符“From”的空行来辨认出消息边界的，请注意后缀的空格。在这则消息投递之前，这一行并不存在，它不同于“From”信头行。许多邮件阅读器不会显示这一行，因此您可能根本看不到它。

```
2:  Return-Path: eric@knecht.Neophilic.COM
```

第 2 行指定一条返回路径，它可能与邮件信头中稍后的 From 行上显示的地址不一样。出错信息应该发送到 Return-Path 信头行中的地址，里面信封上有发件人地址。

```
3:  Delivery-Date: Mon, 06 Aug  2001  14:31:07 -0600
```

第 3 行显示了邮件投递到 Evi 的本地邮箱中的日期。它包括从本地时区（MDT，美国山地时区时间）到 UTC 的偏移量。

```
4: Received: from anchor.cs.colorado.edu (root@anchor.cs.colorado.edu
    [128.138.242.1]) by rupertsberg.cs.colorado.edu (8.10.1/8.10.1) with ESMTP
    id f76KV7J25997 for <evi@rupertsberg.cs.colorado.edu>; Mon, 6 Aug 2001
    14:31:07 -0600 (MDT)
5: Received: from mroe.cs.colorado.edu (IDENT:root@mroe.cs.colorado.edu
    [128.138.243.151]) by anchor.cs.colorado.edu (8.10.1/8.10.1) with ESMTP id
    f76KV6418006 for <evi@anchor.cs.colorado.edu>; Mon, 6 Aug 2001 14:31:06
    -0600 (MDT)
6: Received: from knecht.Neophilic.COM (knecht.sendmail.org [209.31.233.176])
    by mroe.cs.colorado.edu (8.10.1/8.10.1) with ESMTP id f76KV5Q17625 for
    <evi@anchor.cs.colorado.edu>; Mon, 6 Aug 2001 14:31:05 -0600 (MDT)
7: Received: from knecht.Neophilic.COM (localhost.Neophilic.COM [127.0.0.1])
    by knecht.Neophilic.COM (8.12.0.Beta16/8.12.0.Beta17) with ESMTP id
    f76KUufp084340 for <evi@anchor.cs.colorado.edu>; Mon, 6 Aug 2001 13:30:
    56 -0700 (PDT)
```

第 4~7 行记录下这则消息在到用户邮箱的路上经过的各种系统的通道。每台处理邮件消息的机器都会在消息的信头中添加一个 Received 行。新的行添加在顶端，所以在阅读它们时，您是从收件人到发件人跟踪消息的。如果正在查看的消息是一则垃圾邮件，那么您真正可以相信的唯一一个 Received 行就是由您本地计算机生成的那行。

每个 Received 行包含发送它的机器的名称、接收它的机器的名称、接收机器上 **sendmail**（或使用的任何传输代理）的版本、接收机器上这则消息的唯一标识号、收件人（如果只有一项的话）、日期和时间、最后是本地时区与 UTC（Universal Coordinated Time，世界协调时间，以前叫做 GMT，即 Greenwich Mean Time，格林威治标准时间）的偏移量。这一数据是从 **sendmail** 的内部宏变量那里收集的。在接下来的几段中，我们将从发件人到收件人跟踪这则消息（从信头行的角度说，就是反过来看）。

第 7 行说明这则消息从 knecht 的 localhost 接口（Eric[译者注：这是 sendmail 的作者 Eric Allman]写的特殊邮件用户代理选择这个接口用于其初始连接）通过内核的伪设备 loopback 到达 knecht 的外部接口。第 6 行记录的是：knecht 随后将这则消息发向 mroe.cs.colorado.edu，虽然消息的地址是 evi@anchor.cs.colorado.edu（请参见信头第 9 行）。用 **nslookup** 或 **dig** 快速检查一下可以看到，主机 anchor 有一个 MX 记录指向 mroe，正是它使得投递被转了方向。knecht 这台机器正在运行的 **sendmail** 版本是 8.12.0Beta16。有关 MX 记录的更多信息请参考 15.7.6 节。

www.TopSage.com

机器 mroe 运行 **sendmail** 的版本是 8.10.1，当消息位于队列中时，这则消息用队列标识 ID 号 f76KV5Q17625 标识出来。随后 mroe 把这则消息按地址（第 5 行）转发给 anchor.cs.colorado.edu，这可能看上去显得有点儿意外。由于 MX 记录的缘故，原来 knecht 发出要到 anchor 的邮件，却转到了 mroe。之所以出现这种明显不一致的现象，原因是 cs.colorado.edu 这个域使用了“分离式 DNS（split DNS）”的配置。从外界能够看到的 anchor 的 MX 记录指向了内部的邮件主控机（mroe）。不过，在 cs.colorado.edu 域的内部看到的则是不同的一个记录。内部版本的记录首先指向 anchor 自己，然后把 mroe 作为一个备份。

一旦邮件到达了 anchor，它就立即被再次转发，这一次是转发给 rupertsberg。这一跳的原因是别名机制，这种处理邮件的功能将在 18.4 节详细介绍。

在邮件的传输过程中，别名扮演了一个重要的角色。一个别名将一个用户名映射到别的名字上，例如，映射到另一台机器上的同一个用户，映射到一组用户，甚至映射到这个用户的名字的另一种拼写上。您不能只研究例子中的信头就能确定消息为什么发生转向。和 MX 记录一样，您必须寻找外部的信息源。

第 5 和第 6 的 Received 行包含了“for <evi@anchor.cs.colorado.edu>”这一段，它标明了在邮件到达本地站点时是怎样寻址的。当您试图取消订阅邮递列表，而它要求您要么从订阅邮递列表（有时可能是好几年前）的那台主机上发送取消订阅的消息，要么知道哪个地址，在您取消订阅的消息里把它当作个参数来用，这时候有这样的信息就有很大帮助。

最后一个 Received 行（第 4 行）显示“for <evi@mroe.cs.colorado.edu”，**sendmail** 的目的地址宏的值已经被机器 anchor 上查找到的别名给修改了。rupertsberg 上的本地邮件投递代理将邮件发进 Evi 的邮箱。

```
8:  Message-Id: <200108062030.f76KUufp084340@knecht.Neophilic.COM>
```

第 8 行包含的是消息 ID，这个 ID 不同于队列 ID，它在世界范围的邮件系统内是唯一的。它是在消息最初提交给邮件系统时被添加到消息中的。

```
9: To: evi@anchor.cs.colorado.edu
10: From: Eric Allman <eric@Sendmail.ORG>
11: X-URL: http://WWW.Sendmail.ORG/~eric
12: Subject: example message for Evi
13: Date: Mon, 06 Aug 2001 13:30:56 -0700
14: Sender: eric@knecht.Neophilic.COM
```

第 9、10、12、13 和 14 行是标准的。虽然可以不用 Subject 信头，但是大多数用户代理都包含它。To 行包含主要收件人或所有收件人的地址。From 行列出的发送方是 eric@sendmail.org，不过，Received 行列出的发送方机器是位于 neophilic.com 这个域内——除了 sendmail.org 之外，Eric 的机器 knecht 还绑定了几个虚拟域名。

Date 行显示消息发送时的日期和时间。在本例中，发送时间与 Received 行中的日期很接近，尽管它们两者是用不同的时钟测量的。

第 11 行列出了 Eric 主页的 URL。注意，它以一个 X 开头，说明这一行不是正式的信头内容。在首次确定邮件的时候，没有 Web 地址或者 URL 这样的内容。

Received 行通常由传输代理添加（除非它们是伪造的），其他信头由用户代理添加。某些用户代理功能不完整，无法添加正确的信头，此时，**sendmail** 会插手帮助添加丢失的信头。

第一个添加的 Received 行（通常是在发送机器上，当邮件发送到待发接口时添加）有时会包含一个给出发件人登录名的“indent”子句。它应该与 From 行上的名字相同，但如果 From 行是伪造的，那么就会不一样。在我们举的例子里，Eric 的机器 knecht 没有运行实现这一功能（**identd**）的守护进程，所以没有列出发件人登录名。

www.TopSage.com

图 18.2 图解说明了这则消息通过邮件系统的旅程。它显示了采取哪些处理动作、处理是在哪里发生的、以及是由什么程序执行它们的。

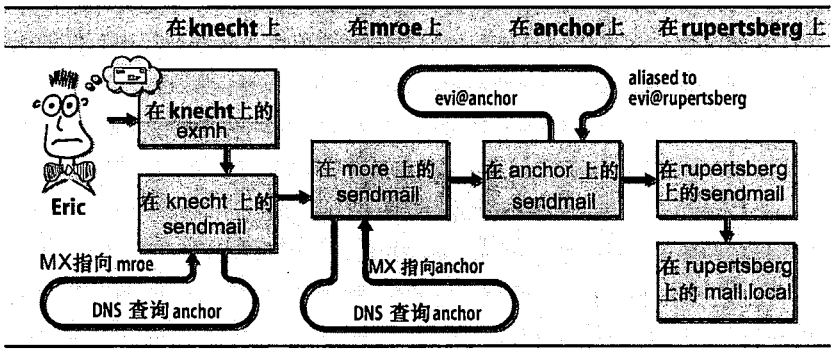


图 18.2 来自 Eric 的一则消息

由此可见，在这一过程中 **sendmail** 是骨干。从消息离开 Berkeley 的 **exmh** 直至它被投递到 rupertsberg，都是由 **sendmail** 处理的。

18.3 邮件基本原理

我们在本章中概述的邮件原理，对于保持中型和大型站点的可管理性而言，几乎一定要遵守。不过，它对小站点也适用。所谓易于管理的主要概念是：

- 有用于传入和传出邮件的服务器，对于真正的大型站点而言，则是一个分层结构；
- 一个物理网点上每个用户有邮件之家（mail home）；
- 用 IMAP 或 POP[7]把 PC、Mac 和远程客户机集成起来。

我们将在下面逐个讨论这些关键问题，然后给出几个例子。其他子系统同样必须与邮件系统的设计相配合：DNS 的 MX 记录必须设置正确、Internet 防火墙必须允许邮件进出、保存消息的机器必须被确定等。有关 MX 记录的更多信息请参见 15.7.6 节。

邮件服务器有 4 个功能：

- 接受从用户代理来的传出邮件，并把它发送到邮件系统中；
- 从外部世界接收传入的邮件；
- 将邮件递送到最终用户的邮箱；
- 让用户通过 IMAP 或 POP 访问他们的邮箱。

在一个小网点上，实现这些功能的服务器可能全都是同一台机器，只是给它具有各式各样的头衔而已。在比较大的网点上，不同的头衔应该是不同的机器。如果传入的邮件只到达一台机器，而传出邮件看起来又只从另一台机器上发出，那么配置网络防火墙的规则就会容易得多。

有些站点使用代理（proxy）接收从外界来的邮件。代理并不真正处理邮件，它只是接收它并把它放入缓冲区。一个单独的进程随后会把缓冲区中的邮件发给 **sendmail** 以便进行传送和处理。www.obtuse.com 的 **smtpd** 和 **smtpfwdd** 就是用于 **sendmail** 的这类代理的例子，**smtpd** 也可以根据访问列表过滤传入邮件。两者都是开放源代码的产品。我们举例的 Linux 发行版本都没有将它们包括在标准安装包中。

[7] 最近 IMAP 比 POP 更值得采用。如果您支持从单位之外访问电子邮件，那么务必使用这两种协议的 SSL 加密版（分别为 IMAPS 和 POPS）。参考 18.3.3 节了解更详细的信息。

www.TopSage.com

TopSage.com

18.3.1 使用邮件服务器

要挑选稳定的、可靠的机器用作邮件服务器。我们在这里概述了一个邮件系统的设计，它看起来大小适当，并且管理和保护起来也相当容易。它把专用于处理传入邮件和传出邮件的服务器的功能集中了起来。图 18.3 图解说明了该系统的一种形式。

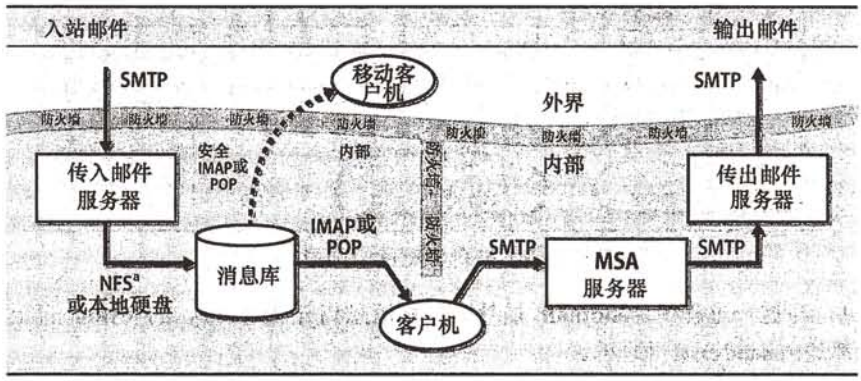


图 18.3 邮件系统的体系结构

a. 我们并不建议将 NFS 用于消息库，因为可能会发生上锁问题。

图 18.3 中描绘的邮件系统对外界只显示为一点：从 Internet 接收消息的邮件服务器。传出邮件服务器也直接连接到了 Internet 上，但由于是它发起连接而不是接受从外部站点来的连接，所以可能没有那么明显。传入邮件服务器应该得到认真的监控，应该用安全补丁对它进行升级，还应该运行 **sendmail** 的最新版本过滤传入邮件中的垃圾。

处理传出邮件的服务器也必须精心维护。它可以包含自己的垃圾邮件过滤器，以便对本地用户进行验证，使他们不会掺和到垃圾邮件问题中去。如果担心您的站点有信息泄漏，那么建立一个服务器，让所有传出邮件都必须通过它传递，这样可以更容易实现或加强内容管制。如果站点要管理很大的邮递列表，可以配置传出邮件服务器使其利用 **sendmail** 某些面向性能的功能，参考 18.13 节了解详情。

如果邮件负荷量大到需要有多台传入和传出邮件服务器的话，这样做也是可以的。例如，多台传入邮件服务器要藏在一台负载均衡的设备后面或者 DNS MX 记录粗略地平衡一下负载。不同的客户端机器可以通过不同的传出服务器发送邮件。但是，不要在传入服务器和传出服务器之间直接传递任何邮件，应该用一堵内部防火墙将它们彼此分隔。

在确实非常大的站点上，都会有重复的多台传入和传出邮件服务器。在查找用户邮箱并把邮件送到合适的消息库上，可以多加一层路由功能。有路由功能的这一层也可以在把消息投递到用户的邮箱之前过滤垃圾邮件和病毒。

为客户设计邮件系统的 ISP 应该添加另一台服务器，让它充当客户备份 MX 记录的目标主机，并且处理邮递列表。这台机器接受邮件并中转回去，但必须对邮件进行严格过滤，以确保只中转真实客户的邮件。也应该用防火墙将它与传入和传出邮件服务器分隔开。

普通的 Linux 主机可以给它一个最小的 **sendmail** 配置，将传出邮件转发给服务器处理。它们不需要从 Internet 接收邮件。有些站点可能要放宽松一点，允许任意主机直接向 Internet 发送邮件。在这两种情况下，不是服务器的机器都可以分享相同的 **sendmail** 配置。您可能要用像 **rdist** 或 **rsync** 这样的工具来分发配置。有关文件分发问题的讨论请参见 17.3 节。

要使用像 Microsoft Exchange 和 Lotus Notes 这样的软件，但又不愿意把这些应用程序直接暴露给 Internet 的站点，可以使用图 18.4 中所示的设计。

无论您选择哪种设计方案，都要确保您的 **sendmail** 配置、DNS 的 MX 记录、以及防火墙规则在

www.TopSage.com

关于邮件方面都实现了相同的策略。

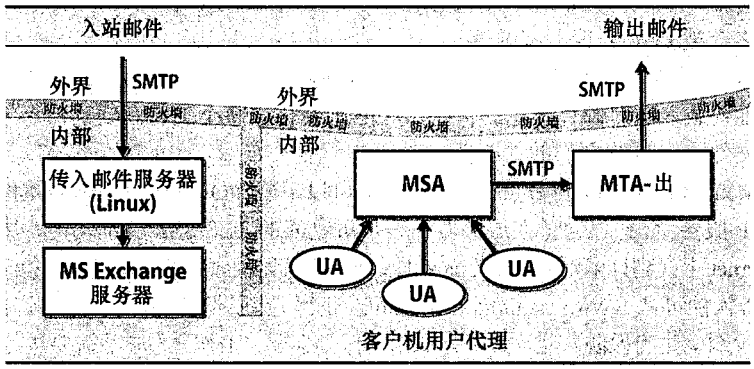


图 18.4 邮件系统的第 2 种体系结构

18.3.2 使用邮件之家

用户可以很方便地在单独的某台机器上接收和保存他们的邮件，即使他们希望从好几种不同的系统访问那些邮件也不难。可以通过 **aliases** 文件或者一个 LDAP 数据库（参见 18.5.2 节）来运行邮件之家（mail home）。用 IMAP 或 POP 可以从远程访问各个用户的邮箱。

我们使用的别名方案允许一个管理域内所有机器上的别名文件相同。这种一致性从管理员的角度看颇受好评（假定登录名和 UID 在所有机器当中是唯一的，这是我们强烈推荐的一种策略）。

有些站点通过 NFS 导出**/var/spool/mail** 来实现邮件的集中。NFS 文件带来的上锁问题会导致用户丢失邮件或弄混他们缓冲区中的文件。找出 NFS、**sendmail** 和投递代理中是谁真正加上的锁，对邮箱被破坏了的可怜用户毫无帮助（不过，**sendmail** 永远是清白的，因为它从不真正投递邮件）。

有些 NFS 的实现（比如在专用 NFS 文件服务器上的那些）带有一个工作良好的上锁管理器。大多数实现或者不支持上锁，或者支持不当。有些站点只是忽略了上锁问题并抱着乐观态度，而其他站点则要求用户在文件服务器上读取邮件。

我们的忠告是根本不要使用通过 NFS 共享的**/var/spool/mail**。

18.3.3 使用 IMAP 或 POP

IMAP 和 POP 是用户的桌面机器连接到网络中时用来下载电子邮件的协议。这是管理邮件的理想方法，对于并不总是与网络连接的主机（不是因为不用时就要被关掉的主机，就是因为在家中要和孩子们分享一条电话线）来说，更是方便。使用它们的时候要求有口令，确保使用一个采用 SSL 加密的协议版本（IMAPS 和 POPS），因而不会在 Internet 上以明文传送口令。

我们喜欢 IMAP（Internet Message Access Protocol，Internet 消息访问协议）的程度胜过了 POP。它投递邮件时每次只投递一条消息而不是发送全部，这对网络要友善许多（特别是在一条慢速链路上时），并且对到处漫游的人来说也更好用。对付一些人发送的巨大的附件来说，它特别棒：您可以浏览邮件消息的信头而不下载附件，直到您已准备好接收它们为止。

IMAP 可以在多个站点之间管理邮件文件夹，例如，在邮件服务器和您的 PC 之间。停留在 Linux 服务器上的邮件可以是正规备份计划的一部分。www.imap.org 包含有关 IMAP 的大量信息和一份有用的列表。

POP（Post Office Protocol，邮局协议）与 IMAP 相似，但它假定了一个模型，在这个模型中所有邮件都是从服务器下载到 PC 的。这些邮件可以从服务器上删除（在此情况下就不能备份了）或者保

www.TopSage.com

存在服务器上（在此情况下您的邮件缓冲区文件就会变得越来越大）。“一次下载整个邮箱”的形式对网络过于苛刻，对用户又缺少灵活性。如果您有一个很大的邮件缓冲区文件，那它在拨号线路上真的会很慢。邮件最终会被 POP 弄得七零八落。

如果用户从来都不删除任何邮件消息的话，这两种协议都可以变成消耗资源的大户。在采用 IMAP 的情况下，它永远要载入所有邮件消息的信头，POP 则会传送整个邮箱。要确保您的用户理解了删除消息或者在本地文件夹中过滤它们的重要性。

从位于 www.eudora.com/qpopper 的 Qualcomm 可以下载 POP3 协议当前版本的一个不错实现。**qpopper** 是个 POP 服务器，它包含了在服务器和客户机之间的 TLS/SSL 验证功能，并且加密邮件消息。

您可以在 Internet 上找到许多别的用于 Linux 的 POP3 服务器程序，务必选择其中得到积极维护的一种。

IMAP 服务器软件可从 www.washington.edu/imap 获得。除了把适当的 IMAP 项加入到**/etc/services** 和**/etc/inetd.conf** 文件，并确保防火墙（如果有的话）不会妨碍它工作之外，不需要进行任何配置。IMAP 在过去有过安全问题的记录，请参考 CERT 的报告并务必得到最新的版本，如果针对您的发行版本有安全报告的话特别要注意。

卡内基梅隆大学（Carnegie Mellon University）开发了一种叫做 Cyrus IMAP 的 IMAP 服务器，它也支持 POP 协议。比起 UW（译者注：华盛顿大学，University of Washington）的 IMAP 实现来，我们更喜欢前者。

Dovecot 是更新的一种软件包，它实现了 IMAP 和 POP 服务。它按照严格和明确的编码规范来编写代码，所以至少从理论上说，提高了它的安全性。Dovecot 还有一些有意思的功能，例如能够把电子邮件保存在 SQL 数据库而不是文件系统中。Dovecot 尚未取得 Cyrus 那样的业绩和使用基础，但它肯定是一个值得关注和评估的项目。

我们举例的所有 Linux 发行版本都带有叫做 **imapd** 的 IMAP 服务器，还有一个客户端 **fetchmail**，它既支持 IMAP 协议也支持 POP 协议。Red Hat 的 **imapd** 是 CMU 的 Cyrus IMAP 服务器，SUSE、Debian 和 Ubuntu 使用 UW 的版本。Red Hat 也带有 **pop 3d**，它是 POP 服务器。SUSE 包含 3 种 POP 服务器（这可不是多余）：**qpopper**（SUSE 把它叫做 **popper**），**pop2d** 和 **pop3d**。Debian 有几种用 IMAP 管理邮箱的工具，命令 **man -kimap** 会告诉您它们的名字。

18.4 邮件别名

别名（alias）能够让系统管理员或单个用户重新发送邮件[8]。他们可以定义邮递列表、在机器之间转发邮件，或者允许用多个名字指定一个用户。别名处理是递归的，所以一个别名指向的其他目的地也可以是别名。

sendmail 支持好几种别名机制：

- 用户和系统管理员很容易建立的各种纯文件的映射，诸如**/etc/aliases**；
- 厂商主推的老式文件发布系统，诸如 Sun[9]的 NIS 和 NIS+，以及苹果公司的 NetInfo；
- 各种邮件路由数据库；
- LDAP（Lightweight Directory Access Protocol，轻量级目录访问协议）。

像**/etc/aliases** 这样的纯文件（本节后面讨论）是迄今为止最简单和最容易在中小站点上建立的别名机制。如果您希望使用邮件之家的概念而且您有一个大型的复杂网点，我们建议您通过把别名保存

[8] 从技术上讲，别名只能由系统管理员配置。用户使用**.forward** 文件来控制邮件路由并不是真正的别名，但我们在这里把它们不加区别地合在一起了。

[9] Sun 于 2005 年正式停止了对 NIS+的支持。LDAP 是指定的替代技术。

www.TopSage.com

在一台 LDAP 服务器上来实现它。

我们已在第 17 章中提到过 LDAP，还会在本章的 3 个地方更详细地讨论它。我们将在 18.5.2 节介绍和阐述 LDAP，然后在 18.9.10 节描述 LDAP 与 **sendmail** 及其配置文件的配合使用，不过，在深入讨论 LDAP 之前，我们先要介绍一下传统的纯文件别名机制。

传统上可以在下列 3 个地方定义别名（遗憾的是，要用 3 种不同的句法）：

- 在用户代理的配置文件中（由发送方用户定义）；
- 在系统范围内起作用的**/etc/mail/aliases** 文件中（由系统管理员定义）；
- 在用户的转发文件**~/.forward** 中（由接收方用户定义）[10]。

用户代理在该用户的配置文件中查找别名，并在把消息送入到邮件系统之前扩展这些别名。传输代理 **sendmail** 在全局的 **aliases** 文件中寻找别名，然后再到收件人的转发文件中寻找。别名只会应用到 **sendmail** 认为是本地的那些消息上[11]。

下面是 **aliases** 文件格式中一些别名的例子：

```
nemeth:  evi
evi: evi@mailhub
authors: evi,garth,trent
```

第一行说的是发送给“nemeth”的邮件应该投递给本地计算机上的用户 evi。第二行说的是所有给 evi 的邮件都应该投递到机器 mailhub，而第 3 行说地址是“authors”的邮件应该投递给用户 evi、garth 和 trent。因为别名支持递归，所以发送给“nemeth”的邮件实际上最后去了“evi@mailhub”。

到全局 **aliases** 文件的路径是在 **sendmail** 的配置文件中指定的——在 Red Hat、SUSE 和 Ubuntu 上都是**/etc/aliases**。**/etc/mail/aliases** 实际上才是“标准”位置。站点可以有多个 **aliases** 文件，且还可以使用别的方法，比如用 NIS 或数据库文件来保存别名映射关系。有关 NIS 的更多信息请参考第 17 章。

aliases 文件中配置项的格式是：

```
local-name:  recipient1,recipient2,...
```

其中 *local-name* 匹配传入消息的源地址，而收件人名单则包含收件人地址或其他别名的名字。缩行被当作是前面行的续行。

从邮件的角度来看，**aliases** 文件代替了**/etc/passwd**，所以：

```
david:  david@somewhere-else.edu
```

将会阻止本地用户 david 得到任何邮件。因此，管理员和 **adduser** 工具在选择新用户的名字时应该把 **passwd** 文件和 **aliases** 文件都检查一下。

/etc/mail/aliases 文件应该永远包含一个叫做“postmaster”的别名，它把邮件转发给负责维护邮件系统的人。类似地，要有一个别名“abuse”，适用于外单位的人需要联络您报告源自于您的网点的垃圾邮件或者可疑网络行为。还必须提供一个接收来自 **sendmail** 的自动消息的别名，通常把它称为 Mailer-Daemon（邮寄程序守护进程），并且经常让它的别名也是 postmaster。

应该把 root 的邮件重定向给站点管理员或者每天都登录进入系统的人。bin、sys、daemon、nobody 和 hostmaster 账号（以及设置的任何其他伪用户账号）也应该拥有把邮件转发给某个人的别名。对于理应包含的在系统范围内起作用的别名，软件发布中的文件 **sendmail/aliases** 是一个很好的模板。它还包含安全性建议和一个例子，说明一些常见的用户请求在伯克利（译者注：**sendmail** 的作者 Eric Allman 是加州大学伯克利分校的毕业生，也在那里工作）那里是怎样发送的。

[10] **~/.forward 是 sendmail** 默认查看的地方。不过，您可以通过设置 **sendmail** 的 ForwardPath 选项来覆盖这个路径。

[11] 实际上，这并不全对。如果您在 SMTP 邮寄程序中加入了标志 F=A，就能实现远程地址的别名机制。

www.TopSage.com

sendmail 通过计算消息信头中 Received 行数来检测导致邮件没完没了来回转发的循环，并在计数值达到某个预设界限时（通常是 25）[12]把邮件返回给发件人。在 **sendmail** 的专门术语中，每次访问到一台新的机器称作一“跳（hop）”，把一则消息返回给发件人称为“弹回（bouncing）”它。前面的表述如果用适当的术语讲，就是“邮件在 25 跳之后弹回”[13]。

除了用户列表，别名还可以指：

- 一个包含地址列表的文件；
- 一个应该把消息附加到其后的文件；
- 一条应该把消息作为其输入的命令。

因为一则消息的内容完全由它的发件人确定，所以这些投递目标常常受到黑客的攻击。**sendmail** 对与这类文件和命令有关的所有权和权限已经变得非常挑剔。要让 **sendmail** 不那么偏执，必须设置 `DontBlameSendmail` 选项中的某一项，之所以叫这个名字是因为不鼓励您这样做。很遗憾，**sendmail** 在遇到不安全的权限或所有权时产生的出错信息并不总是很清楚明了。

18.4.1 从文件中获取邮递列表

`include:`指令是让用户管理他们自己的邮递列表的一种好方法。它允许从外部文件获取一个成员的别名，而不是直接在 **aliases** 文件中列出。还可以在本地修改该文件而不需要负责全局 **aliases** 文件的系统管理员介入。

在设置列表时，系统管理员必须把别名输入全局 **aliases** 文件中、创建被包含文件，并用 **chown** 命令把被包含的文件的所有权赋予维护邮递列表的用户。例如，**aliases** 文件可能包含：

```
sabook:  :include:/usr/local/mail/lah.readers
```

文件 **lah.readers** 应该位于本地文件系统而不是在通过 NFS 安装的文件系统中[14]，并且只应对其所有者是可写入的。为了做到真正完整，我们还应该包含邮递列表所有者的别名，以便将错误（弹回）发给列表的所有者而不是该地址列表的消息的发件人。

```
owner-sabook:  evi
```

有关邮递列表以及它们和 **aliases** 文件交互作用的更多信息，可参考 17.4.6 节。

18.4.2 发邮件给文件

如果别名的目标是一个绝对路径名（如果它包含特殊字符则要用双引号引起来），消息将被附加到指定的文件之后。而该文件必须已经存在。例如：

```
complaints:  /dev/null
```

能够把邮件发送给文件和程序是很有用的，但这一功能也引入了安全性问题，因而受到限制。这一句法只在 **aliases** 文件和用户的 **.forward** 文件（或是用`:include:`插入到这些文件中的某一个文件）中有效。文件名不会被理解成一个标准的地址，所以地址是/etc/passwd@host.domain 的邮件将被弹回。

[12] 默认的跳数上限是 25 次，但你可以在配置文件中修改它。

[13] 我们在本章中的术语有不一致的地方，有时把一则返回消息称为“弹回”，有时则称它为一个“错误”。我们实际上的意思是说产生了一个 DSN（Delivery Status Notification，投递状态通知）。这样的通知往往意味着一则消息因为不可投递而被退回给发件人。

[14] 如果 NFS 文件系统是“硬（hard）”安装的，那么当 NFS 出现故障时，**sendmail** 将发生阻塞，随之而来的是若干打开的文件句柄和一些等待中的进程。最终你可能耗尽进程 ID 或文件句柄，从而不得不重新启动机器以便进行清理。

www.TopSage.com

大家网 TopSage.com

有些用户代理允许您把邮件发给一个本地文件（比如发件箱文件夹）。但只不过是这则消息的副本由用户代理保存起来了，永远不会真正通过邮件系统发送出去。

如果在 **aliases** 文件中引用了目的地文件，那么它必须对任何人可写（不建议这样）、有 setuid 位但不可执行、或者由 **sendmail** 的默认用户所有。默认用户的身份用 `DefaultUser` 选项设置。它通常为 mailnull、sendmail、daemon，或者 UID 1、GID 1。

如果在**.forward** 文件中引用该文件，那么它必须由原来的消息收件人所有，且有对它的写权，而且此收件人必须是一个合法用户，在**/etc/passwd** 文件中有他的一条记录，并且在**/etc/shells** 中列出了他用的合法 shell。对于由 root 用户拥有的文件，请使用模式 4644 或者模式 4600 来设置 setuid 位，但不可执行。

18.4.3 发邮件给程序

别名还可以把邮件发送到一个程序的标准输入。这一做法用下面这样的一行来指定：

```
autoftp:  "|/usr/local/bin/ftpserver"
```

用这项功能甚至比发邮件给文件更容易造成安全性漏洞，所以再强调一次，它只允许用在 **aliases**、**.forward** 或 `:include:` 文件中，**sendmail** 的默认配置现在要求使用受限制的 shell——**smrsh**[15]。在 **aliases** 文件中，程序作为 **sendmail** 的默认用户运行；否则，程序作为**.forward** 或 `:include:` 文件的所有者运行。这个用户必须已经在**/etc/passwd** 文件中列出，并具有一个有效的 shell（在**/etc/shells** 中）。

程序的邮寄程序在运行接收邮件的命令之前先把它的工作目录变为用户的主目录（或者，如果那个目录不能访问，就变为 root 的目录）。默认原本是 **sendmail** 的邮件队列目录，但是有些基于 **csh** 的 shell 不接受。

18.4.4 别名举例

下面是系统管理员可能使用的一些典型别名。

```
# Required aliases[16]
postmaster: trouble, evi
postmistress: postmaster
MAILER-DAEMON: postmaster
hostmaster: trent
abuse: postmaster
webmaster: trouble, trent
root: trouble, trent

# include for local trouble alias
trouble: :include:/usr/local/mail/trouble.alias
troubletrap: "/usr/local/mail/logs/troublemail"
tmr: troubletrap,:include:/usr/local/mail/tmr.alias

# sysadmin conveniences
diary: "/usr/local/admin/diary"
info: "|/usr/local/bin/sendinfo"

# class aliases that change every semester
sa-class: real-sa-class@nag
real-sa-class: :include:/usr/local/adm/sa-class.list
```

[15] 发邮件给程序是一种主要的潜在安全漏洞。参考 18.13 节了解有关 **smrsh** 的更多信息。

[16] 说了个善意的谎言。（根据 RFC）真正只要求有 postmaster 和 MAILER-DAEMON，但是习惯上还是包括了 hostmaster、abuse 和 webmaster。

www.TopSage.com

在这个例子中，我们希望来自校园各处的用户能在发生问题时向一个别名“trouble”发送邮件。问题报告一定要传递给一个适当的本地系统管理员小组。特别地，我们愿意设置邮件别名是为了：

- 让问题邮件总是去到一个适当的分组；
- 在所有主机上使用同一个简单版本的别名文件；
- 让单个管理小组控制他们自己的分发列表；
- 把所有问题邮件的副本发往各个小组所用的一个本地日志文件。

通过从每台机器上的一个文件中来获取 trouble 别名的定义，上述配置可以满足这些目标。发送给地址 trouble@anchor 和 trouble@boulder 的邮件最终会发到不同的地方，即使 anchor 和 boulder 使用同样的**/etc/mail/aliases** 文件。

问题邮件通常在各个地点的某台特定机器上处理。例如，在一台从（slave）机器上的 **trouble.alias** 文件可以包含地址：

```
trouble@master
```

以便将问题邮件送往适当的主机器。

在分析一则 trouble 消息时，它被发送给别名“tmr”，此别名代表“trouble mail readers（问题邮件阅读器）”。tmr 别名把这则消息归档到 troubletrap 别名，还把它发送给一个用户列表，这个列表是从主（master）机器上的一个文件中获得的。把管理员新手添加到 tmr 列表中是一种很棒的方法，这可以让他们看到出现的需要支持的问题、管理员的答复，还有对用户（也就是客户）应该适当使用的奉承语调。

别名 sa-class 有两个层次，这样一来，只需要在单台机器 nag 上维护包含学生列表的数据文件。前面 18.4.1 节里 sabook 别名的例子应该说真正具有了与这种类型相同的间接手段，因此不需要复制包含文件。

别名 diary 很方便，对于不耐烦记录下自己所做事情的乖僻的学生系统管理员来说，它很好地发挥了作为一种文档提取技术的作用。通过发送邮件给 diary 文件，系统管理员很容易就可以记录下机器生命周期中的重要事件（OS 升级、硬件改动、崩溃等）。不要把这个文件放在包含日志文件的文件系统中，那样做会让黑客填满文件系统而使 syslog 无法写入日志条目（也就掩盖了他们的踪迹）。

18.4.5 邮件转发

aliases 文件是一个系统范围的配置文件，它应该由管理员来维护。如果用户希望重新指定发送自己邮件的路径（而您的站点不使用 POP 或 IMAP 来访问邮件），可以在他们自己的主目录中创建**.forward** 文件来实现这一点。以前，**sendmail** 总是在用户的主目录中搜索一个**.forward** 文件，但现在要启用这种做法，则必须设置 ForwardPath 变量，覆盖默认的设置。当用户希望在一台特定主机上接收邮件，或当某人离开您的网点而希望让邮件转发到一个新位置时，使用**.forward** 文件会很方便。

一个**.forward** 文件由单独一行上用逗号分隔的地址列表，或多行上的若干项组成。例如：

```
evi@ipn.caida.org
evi@atrust.com
```

或者

```
\mcbryan, "/home/mcbryan/archive", mcbryan@f1supi1.gmd.de
```

在第一个例子中，evi 的邮件不是投递到本地计算机上，而是转发给加州圣地亚哥 CAIDA 的机器 ipn 和 evi@atrust.com。第二个例子来自一个不信任邮件系统的用户，他希望把自己的邮件复制到 3 个地方：本地计算机上的常规邮件缓冲区、所有传入邮件的一个永久存档文件、他此刻游历的德国国内

www.TopSage.com

的一个临时地址。他的用户名之前的反斜线的意思是，无论别名或转发文件说什么，都要把邮件投递到本地。

对于邮件路由中的临时变化，使用**.forward** 文件要比使用全局 **aliases** 文件更可取，因为修改系统范围的别名所需要的开销（计算机时间和人力时间）是相当高的。

用户的**.forward** 文件必须由该用户所有，决不能让组成员或任何人都可写入。如果 **sendmail** 认为到**.forward** 文件的目录路径是安全的（也就是说，从 root 用户以下的权限都没有问题），它就能是一个链接，否则，它就不能是一个链接。**sendmail** 将忽略掉权限可疑的转发文件，上级目录的权限也必须是安全的（只能由拥有该文件的用户写入）。

自然，**sendmail** 必须要能够访问邮件所投递到的机器上该用户的主目录，才能确定它是否包含一个**.forward** 文件。地址的永久性修改应该放入**/etc/mail/aliases** 文件，因为用户的主目录和文件最终将被删除。

sendmail 有一项很棒的功能，`FEATURE (backqucte redirect apostrophe)`，它能帮助管理电子邮件的永久性变更。如果一个别名指向 user@newsite.REDIRECT，邮件将被退回给发件人，并附上新地址的通知。消息不会转发到新地址，所以发件人必须更新他的地址簿并重新发送消息。

您可以配置 **sendmail** 以便支持用于**.forward** 文件的中央目录，但用户不能寄期望于这项配置。**.forward** 文件的位置由 `ForwardPath` 选项控制，它通常指向中央目录，然后指向用户的主目录。18.8.3 节中举例说明的 **generic.m4** 域文件包含了用于**.forward** 文件的中央位置的一个例子。

全局 **aliases** 文件中的项要优先于**.forward** 文件中的项。因为这些文件是由不同的人维护的，所以用户必须当心，不要在无意中造成邮件循环。如果用户在网络上拥有一个邮件之家（因此在全局 **aliases** 文件中拥有一项），该用户就不能使用**.forward** 文件把邮件路径重新改到另一台共享同一别名的机器上去。例如，在科罗拉多大学，我们使用一个在本网点范围内起作用的 **aliases** 文件，有类似：

```
evi:  evi@boulder
```

这样的一项，而机器 boulder 上的**.forward** 文件包含：

```
evi@anchor.cs
```

这就造成了一个循环。地址是 evi 的邮件将转发给 boulder，boulder 上的**.forward** 文件又会使得它被发给 cs 子域中的 anchor。anchor 上的 **aliases** 文件使得邮件被转发回 boulder，如此循环不已。在 25 跳之后，邮件将被退回给发件人。

如果您的主要通信模式是电子邮件，那么通知用户发生了邮件循环是很难的事情。发给*\user*[17]的邮件会把消息投递到本地计算机上，而不管系统范围的 **aliases** 文件或用户的**.forward** 文件可能说什么。如果本地计算机是用户要读取邮件的地方，那很好，否则，请给 postmaster 发邮件报告出现了循环，或打个电话！

18.4.6 散列的别名数据库

因为 **aliases** 文件中的项没有特定的顺序，所以 **sendmail** 直接搜索这个文件的效率并不高。取而代之的做法是，用 Berkeley DB 数据库系统或 **ndbm** 数据库系统来构造一个散列的版本。这种散列法可以明显加快别名的查找，特别是在文件变大的时候。

对 DB 而言，从**/etc/mail/aliases** 衍生出的文件名为 **aliases.db**，您每次修改 **aliases** 文件时，都必

[17] 你可能不得不使用两个或更多反斜线来让它们中的某一个通过 shell 进入 **sendmail**。

www.TopSage.com

须用 **newaliases** 命令重建此散列数据库。**newaliases** 其实只是被伪装起来的带命令行标志（-bi）的 **sendmail**，这个标志告诉 **sendmail** 要重建数据库。如果自动运行 **newaliases**，要把出错的输出保存下来——您可能已经引入了格式错误。

18.5 邮递列表和实现清单的软件

邮递列表是一个巨大的别名，它把邮递给它的每则消息的副本发送给加入此列表的所有人。就像是通过电子邮件投递的一个 Usenet 新闻组。有些邮递列表拥有数以千计的收件人。

邮递列表通常在 **aliases** 文件中指定，但保存在某个外部文件中。**sendmail** 和大多数邮递列表软件可以理解一些标准的命名约定。有经验的用户也已经开始依赖它们了。最常用的就是“-request”后缀和“owner-”前缀，用它们来联络邮递列表的维护者。下面这些别名举例说明了这些约定：

```
mylist: :include:/etc/mail/include/mylist
owner-mylist: mylist-request
mylist-request: evi
owner-owner: postmaster
```

在这个例子中，mylist 是邮递列表的名字。邮递列表的成员则从文件**/etc/mail/include/mylist** 中读取。对发到列表并被回复的邮件将发送给邮递列表的所有者 evi，作为加入这个列表的申请。从“owner”到“request”再到“evi”的做法是有用的，因为 owner 的地址（在本例中是 mylist-request）变成了发送给此列表的各个消息的 Return-Path 地址。用 mylist-request 比用这个领域实际维护者的名字要好一点。发送给别名 owner-mylist（其实就是 evi）的消息中的错误将被发给 owner-owner。

消息不可投递的情况称为弹回（bounce）。发送有关弹回的出错信息时无法投递的情况称为二次弹回（double bounce）。于是在我们的例子中，二次弹回被发给 owner-owner 即 postmaster。

如果使用整个站点的别名文件，则需要再多加一级迂回，把 mylist 指向 myreallist@master，以使包含成员列表的数据文件只需要存在于一个地方。

18.5.1 维护邮递列表的软件包

有几种软件包可自动维护邮递列表。它们一般允许用户把自己添加到列表中和从列表中删除自己、获得有关此列表的信息、以及通过电子邮件获得文件。几个流行的邮递列表管理器（以及它们的下载来源）是：

- Majordomo，来自 www.greatcircle.com（SUSE 自带）；
- Mailman，GNU 邮递列表处理器，来自 www.list.org；
- ListProc，来自 www.cren.net；
- SmartList；
- **Listmanager**；
- LISTSERV Lite，来自 www.lsoft.com（商业软件 LISTSERV 的自由版本）。

一般而言，SmartList 小而简单，ListProc 大而复杂，而其他软件介于两者之间。它们的不同之处在于对维护邮递列表的思想不同，其中一些向作为管理者的系统管理员倾斜（ListProc），而另一些则向作为维护者的用户倾斜（Majordomo、Mailman、SmartList、LISTSERV Lite）。Mailman、**Listmanager**、Majordomo 和 LISTSERV Lite 支持远程管理，列表维护甚至不需要在列表所在的机器上拥有登录名，因为所有事务都通过电子邮件进行。大多数的列表软件包允许把邮递给列表的信息汇编成文摘，有些是自动进行的（ListProc、Mailman、**Listmanager** 和 LISTSERV Lite），而有些则通过手工配置（SmartList

www.TopSage.com

和 Majordomo）。

Mailman 是我们最喜欢的列表管理器。它让系统管理员很轻松，而由列表的维护者调整其列表的所有特性。ListProc 和 LISTSERV Lite 是专有软件：前者价钱昂贵，后者则只有二进制文件而且功能又不全。我们没试过 SmartList，但我们喜欢 **procmail**，这要看具体情况了。

我们会在下面简要地描述所有这些软件包。至于更多的细节，请参见各个软件包的相关文章。

一、Majordomo

Majordomo 是一个 Perl/C 软件包，可从 www.greatcircle.com 获得。它最初由 Brent Chapman 编写，Majordomo 现在已经停止开发了，Majordomo 2 是一个完全重写的版本，但似乎在 2004 年就销声匿迹了，所以我们只介绍原来的版本。尽管手册页说它在**/usr/lib/mail/majordomo** 里，可它实际藏在目录**/usr/lib/majordomo** 里。

Majordomo 作为一个无特权的用户运行，典型情况下使用用户名 majordomo 和默认的 daemon 用户组。如果系统支持长用户名（>8 字符），您可以把 majordomo 用作登录名。这个用户必须被 **sendmail** 认为是“可信的”，所以必须在 **sendmail** 配置中提到他，通常是在 `confTRUSTED_USERS` 声明中[18]。有关可信用户的更多信息请参见 18.12 节。

Majordomo 是通过 **majordomo.cf** 文件配置的，这个文件由有效的 Perl 命令组成，这些命令负责初始化变量、定义各个事物所在（或应该在）的目录，指定要支持的列表以及配置被弹回邮件的处理。帮助程序 **config-test** 可以测试配置文件中是否有丢失的变量或错误的句法。SUSE 把配置文件放在**/etc** 下，而把 Majordomo 软件所带的 **config-test** 放在了**/usr/lib/majordomo** 里。

Majordomo 要求 **sendmail** 的 **aliases** 文件中安装有特定的别名。把这些别名集成到一起的最好办法是单独创建一个只用于 Majordomo 的别名文件（**sendmail** 最近的一些版本可以支持多个别名文件）。此文件包含一组 Majordomo 自己使用的别名和一组用于它所管理的各个邮递列表的别名。Majordomo 软件发布中包含一个别名文件的例子：**majordomo.aliases**。

用户有关邮递列表最常见的问题是“我怎样取消订阅？”对于由 Majordomo 管理的列表来说，对 listname@host 的回答是：发送邮件给地址 majordomo@host，在消息的主体内（不是在标题行上）包含“unsubscribe listname”或“unsubscribe listname 电子邮件地址”字样。

如果用第一种形式，则需要从订阅此列表时使用的同一台主机上发送取消订阅的消息；如果用第二种形式，主机应该是电子邮件地址的一部分。即使您已经忘记加入列表时使用的是哪一台机器，也可以从邮件信头中找到这项信息以便正确地取消订阅，如何进行请参见 18.2.2 节的提示。一些邮递列表也接受发送给 listname-request@host 并且在主体内带有关键字“unsubscribe”的邮件。

永远不要把取消订阅的消息发送给列表本身。如果这样做，您的消息将广播给列表中所有的收件人，而您还不知道自己干了什么。

二、Mailman

Mailman 最近才刚刚加入到邮递列表软件的队伍中来（2.1.8 版发表于 2006 年 4 月），可以从 www.list.org 或 GNU 存档中得到它。它最初由 John Viega 编写，目前是由 Ken Manheimer 和 Barry Warsaw 联合开发。和 Majordomo 一样，Mailman 主要是用一种脚本语言编写的，再加上 C 语言的封装，但使用的是 Python 语言（可从 www.python.org 获得）。

Mailman 是其作者们在使用 Majordomo 时，遇到了弹回错误时的挫折，配置高级功能比如文摘和中等规模列表时感到棘手，而且成批邮寄时遇到性能的障碍，于是产生了编写 Mailman 的灵感。Mailman 提供了一个导入 Majordomo 列表的脚本。它也有一些检测和控制垃圾邮件的能力。

Mailman 的好名声源自于其 Web 界面，这个界面让 moderator 或 postmaster 可以很容易地管理列

[18] 可信用户可以改变一则消息信头中的 From 行，可以重新构造 **aliases** 文件。

www.TopSage.com

表，用户也可以很容易地订阅、取消和配置自己的选项。

三、ListProc

ListProc 是邮递列表管理软件中的老前辈。它是在 1991 年由 Anastasios Kotsikonas 编写的，并且一直维护到 1994 年。随后它被搁置了好几年，但它近来因为 1998 年发表的一个新的 beta 版而复兴了。它过去可以从波士顿大学的计算机系免费获得，但其许可证规则有点奇怪。现在可从 www.cren.net 获得它，但使用许可费很高（每份拷贝 2000 美元，甚至对大学也是这个价）。忘记 ListProc 吧，在免费的、开放源代码的软件包里找一个好了。

四、SmartList

SmartList 最初由 Stephen R. van den Berg 编写，他也是 **procmail** 软件包最初的作者。SmartList 可从 ftp.procmail.org 获得。它使用 **procmail**，所以需要把 **pocmail.tar.gz** 和 **SmartList.tar.gz** 都下载下来。或许最简单的办法就是直接下载适用于您系统的 Linux 软件包。

SmartList 小而简单。它是 C 代码、**procmail** 规则和 shell 脚本的组合。弹回——维护邮递列表时头疼的问题——会自动由软件处理。在发给某个用户的邮件被弹回特定次数后，会自动从列表中删除该用户。SmartList 要求在 **passwd** 文件中有一个登录项（“smart”或者可能是“list”），它在 **sendmail** 的配置文件中是一个可信用户。

软件的安装包括 **led**，这是一个用于进行编辑的上锁封装程序（lock wrapper），以防 SmartList 成为一个不连贯的、部分编辑的配置文件。

五、listmanager

listmanager 是由 Murray Kucherawy 用 C 语言写的，所以它要比用 Perl 或者 Python 这样的脚本语言写的软件速度快。**listmanager** 也通过使用 DB 数据库软件包，而不是纯文件和文件系统来提高速度。列举其功能的清单有一页长，其中包括 Web 界面、摘要和几种在安全方面的增强功能。

list manager 确实像是个列表管理器的杀手锏——唯一的缺点是源代码还没有发布出来。按照 Web 网页上的说法，代码在作者完成一遍清理工作、避免可能造成的尴尬之后才会发布。现在可以得到 Linux 的二进制程序。

六、LISTSERV Lite

Eric Thomas 编写的 LISTSERV Lite 是 LISTSERV 的一个残缺版本，商业产品 LISTSERV 来自 L-Soft International，Inc（L-Soft 国际有限公司）。原版软件的一些功能没有了，这个软件限于管理 10 个邮递列表，最多 500 人。LISTSERV Lite 需要作为伪用户 listserv 运行，它必须拥有自己的文件。它还希望有一个 listserv 组。LISTSERV Lite 为订阅列表和维护列表都提供了一个 Web 界面。

它的软件发布可从 www.lsoft.com 获得。源代码不发布，但提供用于许多 UNIX 和 Linux 版本的预编译好的二进制和占位程序。如果您已经熟悉了 LISTSERV 并且有列表在使用它，也许您能够证明运行一个只有二进制程序的、残缺的列表管理器是可行的。如果是从头开始，应该代之以选择上面提到的任意一个无限制的开放源代码产品。

18.5.2 LDAP：轻量级目录访问协议

LDAP 是一种协议，它可提供到一般管理性目录服务的访问。它已经存在好几年了，但只是最近才开始流行。

管理员们正发现 LDAP 擅长许多事情。

- **sendmail** 配置：别名、虚拟域、邮件之家、数据库访问、以及表。
- 用户管理：登录名、口令、主机（例如，斯坦福大学）。
- 管理性配置文件（例如，SUSE）。
- 作为 NIS 的一种替代技术。

www.TopSage.com

- 作为日历服务器。
- 与 PAM（Pluggable Authentication Modules，可插入式身份验证模块）一起使用。

可以想象，LDAP 最终将成为一种可以用于许多不同目的的全局目录系统。遗憾的是，仍然缺乏自动完成 LDAP 一般性系统管理任务的工具。

LDAP 脱胎于 ISO 协议和 X.500 目录服务。这种继承关系立即使人想起复杂、臃肿、累赘、糟糕等，但 LDAP 中的 L（轻量级）则要求考虑到所有这一切。该协议的第 1 版和第 2 版已经成为标准。第 3 版也已经接近完成。幸运的是，所有版本都是向后兼容的。LDAP 协议第 3 版支持层次结构的服务器，为数据而查询一台服务器可以被引向另一台服务器。第 2 版支持层次结构的数据，但只有第 3 版才支持层次结构的服务器。

邮件别名特别适合于使用 LDAP，尤其是现在 **sendmail** 的内部就支持 LDAP，那么就更适合用它了。**sendmail** 在查询 LDAP 服务器时可以进行别名查找，而不是直接查找它们。LDAP 还可以管理邮件路由和虚拟域。对 LDAP 的支持必须编译到 **sendmail** 的二进制程序中。

如果您在寻找一个 LDAP 实现，我们推荐由 www.openldap.org 的 OpenLDAP 小组所提供的服务器程序。这个小组接管并增强了早期由密执安大学（University of Michigan）开发的一个服务器程序的代码。有关 LDAP 软件的更多信息请参考 17.5 节。

LDAP 数据库的项类似于变量名更长的一个 termcap 项。LDAP 数据库中的属性（变量名）尚未完全标准化，这种实际情况可能会导致不同 LDAP 实现之间的不兼容性。

一个数据库项的第一行上的属性由 LDAP 配置文件定义。本节中的例子假定此 LDAP 服务器守护进程（在 OpenLDAP 中是 **slapd**）是用一个根的区别名（`rootdn`）来配置的：

```
"cn=root, dc=synack, dc=net"
```

`dc` 属性出现了两次，因为域的组成部分不能包含一个点（.），为了表示域 synack.net，就需要用两项。更进一步的属性，或者说变量名，可以是您希望的任何东西。它们不区分大小写。

sendmail（它的代码寻找具体的属性名称并赋予它们预先确定的解释）、LDAP 服务程序和 LDAP 数据库的构造程序必须保持协调，使用相同的命名规则。

出现在一个数据库项（数据库关键字）的第一行的一些属性可能有：`dn` 表示域名，`dc` 表示域的成分，`o` 表示某个机构的名称，`c` 表示国家的名称，而 `uid` 表示唯一的 ID（例如，一个登录名）。

sendmail 使用 LDAP 服务器的方式非常像它使用 DNS 名字服务器的方式。**sendmail** 要求 DNS 服务器把名字解析成 IP 地址，以便发送消息。它访问 LDAP 服务器是为了查出别名，以便可以把消息送到正确的地方。在上述两种情况下，查询操作都从纯文件（**hosts** 和 **aliases**）换到了数据库，而且由服务器来管理查询。

在默认情况下，**sendmail** 认识下列 LDAP 数据标记：

```
mailLocalAddress
mailRoutingAddress
mailHost
```

8.12 版的 **sendmail** 扩大了这项默认的设置（参见 **cf/sendmail.schema**），也能让您定义自己的方案来使用您喜欢的任何 LDAP 标记。

下面的例子是 OpenLDAP 实现中的 **ldap.conf** 文件：

```
# LDAP Defaults, ldap.conf file, should be world-readable.
#
BASE    dc=synack, dc=net
HOST    gw.synack.net
PORT    389
```

www.TopSage.com

它支持以下形式的数据库项：

```
dn: uid=jon, dc=synack, dc=net
objectClass: inetLocalMailRecipient
mailLocalAddress: jon@synack.net
mailRoutingAddress: stabilej@cs.colorado.edu
uid:jon
```

当已经在 **sendmail** 的配置文件中指定过 `FEATURE(ldap_routing)` 的时候，传入邮件的收件人要用 `mailLocalAddress` 字段来匹配。如果匹配，那么邮件就被重定向到 `mailRoutingAddress`。`objectClass` 一行必须要有——它来源于规定 LDAP 和邮件系统交互方式的 RFC 草案。

在主机 gw.synack.net 上，这个数据库项对应别名：

```
jon:  stabilej@cs.colorado.edu
```

有点儿啰嗦，不是吗？这些数据库的项可以代替 **aliases** 文件中的标准配置项，用来给每个用户定义其各自的邮件之家。

有些大单位已经把用户的供应信息（例如每个用户选用的邮箱）保存在另一个数据库里了。如果您的站点上就是这种情况，那么您或许想写一些简短的脚本，提取出这些信息，然后再送入 LDAP。如果您的单位不大，那么可能想坚持采用传统的**/etc/mail/aliases** 文件，或者写一个脚本从命令行生成正确的 LDAP 数据项。

aliases 文件仍然是定义邮递列表（采用`:include:`指令）最好的方法。邮递列表软件一般把消息通过管道送给一个封装脚本（wrapper script），重新发送它。LDAP 查询返回的别名指向`:include:`文件，或者诸如 **majordomo** 这样的程序，但是它不能直接调用一个程序。

至于 8.12 版的 **sendmail**，LDAP 也能用来保存 **sendmail** 的某些其他数据的内容（例如，表[table]和类[class]）。参考 18.9.10 节了解有关配置 sendmail 使用 LDAP 的更多知识。

18.6 sendmail：电子邮件的指挥

我们提到的所有 5 种 Linux 平台都带有一个邮件传输代理。Red Hat、Fedora 和 SUSE 提供了 **sendmail**；Debian 和 Ubuntu 默认包含 Exim。我们会在 18.15 节简要介绍一下 Exim。Exim 的许多组成部分都和 **sendmail** 的类似。既然这一章已经写得很长了，所以我们还是只详细讲解 **sendmail**。我们会拿 **sendmail** 和 Exim 等价的成分来介绍 Exim。

sendmail 之所以能适应标准制订者心血来潮的“奇思妙想”，部分应归功于其配置文件的灵活性，它能让 **sendmail** 满足不同用户群的需要。本章接下来的内容主要帮助读者理解和介绍这个配置文件的结构，这个文件就是“声名狼藉”的 **sendmail.cf**。

sendmail 是一个传输代理，一个在用户代理和投递代理间充当桥梁的程序。它使用 SMTP 协议进行通信，通过 Internet 把消息投递给远程机器上的对等传输代理。**sendmail** 的任务包括：

- 当消息离开用户的键盘时对它们进行控制；
- 理解收件人的地址；
- 选择一个适当的投递或传输代理；
- 把地址重写为投递代理可理解的形式；
- 根据需要重新确定信头的格式；
- 把变换后的消息传递给投递代理。

如果消息无法投递，**sendmail** 还会生成出错消息并把消息返回给发件人。

www.TopSage.com

18.6.1 sendmail 的版本

在撰写本书（英文版）的时候，大多数 Linux 发布商所带的 **sendmail** 版本都源于 V8。不过在通常情况下，它们要比 Sendmail, Inc.（Sendmail 有限公司）的主版本号落后一到两个版本。各厂商常常定制 **sendmail** 的某个特定版本，以后就不太情愿去升级他们的基本系统，以便把当前的修订版本包括进来。

我们把对 **sendmail** 的讨论建立在 8.13 版的基础上，并完全忽略掉 V5 和 IDA，它们已经被废弃了。V8 使用 **m4** 宏指令处理器，在标准应用场合下可以很容易配置好。这个“简装版配置”足以满足大多数站点的需要。

很遗憾，如果您的配置有问题，您可能不得不基于对原始配置文件的了解来进行调试，而我们听到的对这个文件的描述是不易接近、使人畏缩、吹毛求疵、隐晦、繁重、无耻、无聊、虐待狂、不好理解、冗长乏味、荒谬、使人迷乱和扭曲的。我们在本书的老版中对原始配置文件的讲解不多，但是因为它和现代系统管理员的关系已经不那么密切了，所以我们现在推荐您参考 **sendmail** 软件发布中附带的 *Sendmail Installation and Operations Guide* 一文。

有时发布 **sendmail** 的新版本是为了解决安全问题，我们建议您仔细查看 www.sendmail.org 上的版本说明，如果少了与安全性相关的任何一个补丁，则要进行升级。您通常可以从您的 Linux 发行商那里直接获得升级后的 **sendmail** 软件包，但要确保您收到的二进制软件对应当前的 **sendmail** 版本。如果您需要编译和安装从 www.sendmail.org 直接下载的源代码，那么需要一个 C 编译器和 **m4** 宏指令预处理器（它们通常都包括在标准的 Linux 发行版本中）。

有时很难确定真正的 **sendmail** 基本版本，但如果供应厂商还没有把事情搞得更乱，您就可以运行：

```
$ /usr/sbin/sendmail -d0.1 -bt < /dev/null
```

让 **sendmail** 透露其版本、编译到它内部的选项，以及读取配置文件之后它认为自己是谁。**-d** 标志设置了一个调试级别（参考 18.14.1 节了解有关 **sendmail** 中调试级别的更多信息），**-bt** 标志将 **sendmail** 置于地址测试模式，而来自 **/dev/null** 的重定向没有给它提供测试地址。下面是一些输出的例子（稍作过裁减）：

```
Version 8.11.6
 Compiled with: LDAPMAP MAP_REGEX LOG MATCHGECOS MIME7TO8
     MIME8TO7 NAMED_BIND NETINET NETINET6 NETUNIX NEWDB NIS
     QUEUE SASL SCANF SMTP TCPWRAPPERS USERDB
============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = coyote
  (canonical domain name) $j = coyote.toadranch.com
       (subdomain name) $m = toadranch.com
             (node name) $k = coyote.toadranch.com
========================================================
```

sendmail 始终会要用到 DNS MX（mail exchanger，邮件交换器）记录，而且如果是用 NAMED_BIND 选项编译的（如前例所示），它就会这么做。像 $w 这样的单字母变量是从原始配置文件来的，或者是在运行时刻决定的。

18.6.2 从 sendmail.org 安装 sendmail

截至 8.12 版，**sendmail** 的安装环境稍微有点儿变化。**sendmail** 不再 setuid 成 root 来运行，而是 setgid 成 **sendmail** 组——smmsp——来运行。在安装 **sendmail** 之前，您必须创建用户 smmsp 和用户组 smmsp（这个神秘的名字代表 sendmail mail submission program）。我们使用 UID 和 GID 25 来和 SMTP 邮件协议知名的端口号相配。这个 smmsp 用户应该把 smmsp 作为它的默认登录组，这一般在

www.TopSage.com

/etc/passwd 文件中设置。增加一个专门的 sendmail 用户和用户组能够让 **sendmail** 运行的时候没有那么大的特权，从而提高了安全性。

从系统管理员的角度来看，第二个主要的变化是 **sendmail** 现在使用两个配置文件：**submit.cf** 和 **sendmail.cf**。**submit.cf** 文件控制着如何处理本地主机所发出（并且正在提交给邮件系统）的邮件，sendmail.cf 文件控制着传入邮件，以及在提交过程中如何把邮件排队。submit.cf 是由软件发布所提供的，对于所有的站点都一样，通常不需要定制它。

本节简要地描述安装过程。详细情况，以及与特定体系结构或操作系统相关的问题，请参考 **sendmail** 软件发布中的安装说明。下一节介绍在 Debian 系统上用 **apt-get** 安装 **sendmail**。如果您正在替换供应厂商的 **sendmail** 版本，某些配置工作（比如安装帮助文件）可能已经替您完成了。

成员有：

- **sendmail** 二进制程序，通常安装在/usr/sbin 中；

```
-r-xr-sr-x   root  smmsp  /usr/sbin/sendmail
```

- 配置文件**/etc/mail/sendmail.cf** 和**/etc/mail/submit.cf**（在 8.12 及以后的版本中），由系统管理员安装；

```
-rw-r--r--   root  bin  /etc/mail/sendmail.cf
rw-r--r--    root  bin  /etc/mail/submit.cf
```

- 邮件队列目录**/var/spool/mqueue** 和**/var/spool/clientmqueue**（在 8.12 及以后的版本中），由系统管理员或者安装过程创建；

```
drwxrwx---  smmsp   smmsp  /var/spool/clientmqueue
drwx------  root    wheel  /var/spool/mqueue
```

- 到 **sendmail** 的各种链接（**newaliases**、**mailq**、**hoststat** 等）[19]；
- **sendmail** 的更安全的本地投递代理，**smrsh** 和 **mail.local**（通常安装在**/usr/libexec** 中）。

您可以从 www.sendmail.org 下载 **sendmail** 的最新版本。要编译和安装此软件包，可以按照软件目录下的 **INSTALL** 文件中的指导来做。首先添加 smmsp 用户和用户组；不要给这个用户一个实际能用的 shell。下面是一个典型的**/etc/passwd** 配置项：

```
smmsp:x:25:25:Sendmail  Mail  Submission  Prog:/nonexistent:/bin/false
```

下面是一个典型的**/etc/group** 配置项：

```
smmsp:*:25:smmsp
```

要编译这个软件，先进入到软件的 **sendmail** 目录下，运行脚本 **Build**，然后再运行 **Build install**。文件 **devtools/OS/Linux** 包含了在 Linux 系统上构造 **sendmail** 时所做的假定。Linux 发行版本没有对各种东西应该放在何处形成标准，所以 **devtools/OS/Linux** 的内容只是尽力所做的猜测，可能不是您发行版本上真正的情况。

不过，在开始编译之前，您必须确定一种数据库格式和一个 **sendmail** 与管理数据库（比如 NIS）的接口策略。对于磁盘上的数据库，我们推荐 Berkeley DB 软件包，它在 **Makefile** 中指定为 NEWDB（默认情况）。

定制 **Makefile** 的时候不要编辑它，而是创建您自己站点的 **site.config.m4** 文件，把它放入 **devtools/Site** 目录，针对您的操作系统和本地环境来调整它。例如，如果您想要使用 LDAP 和新的垃圾邮件过滤库，那么可以在那个目录里创建 **site.config.m4** 这个文件，其中包含下面这几行：

[19] 这里要小心。有些厂商使用了硬链接，结果当您升级的时候，可能会让 **sendmail** 的版本和 **newaliases** 的版本不匹配，从而造成狡猾且难以发现的头痛问题。

www.TopSage.com

大家网 TopSage.com

```
APPENDDEF('confMAPDEF', '-DLDAPMAP')
APPENDDEF('confLIBS', '-lldap -llber')
APPENDDEF('conf_sendmail_ENVDEF', '-DMILTER')
```

define 语句会替换这个属性当前的定义，APPENDDEF 宏则是追加到当前定义的后面。

编译 **sendmail** 用命令：

```
$ sh ./Build
```

它将自动把站点的特定项包括进来。要把 **sendmail** 安装到正确的位置，运行命令：

```
$ sudo sh ./Build install
```

semdial 一般不应该设置成由 **inetd/xinetd** 来控制，所以它必须在系统引导时由 **rc** 文件显式地启动。一般采用的命令序列类似于：

```
if [-f /usr/sbin/sendmail -a -f /etc/mail/sendmail.cf];
then
    (cd /var/spool/clientmqueue; rm -f [tTx]f*)
    (cd /var/spool/mqueue; rm -f [tTx]f*)
    /usr/sbin/sendmail -bd -q30m     ### queue runner for regular queue
    /usr/sbin/sendmail -Ac -q30m &  ### queue runner for client queue (8.12)
    echo -n ' sendmail' > /dev/console
fi
```

这几行检查 **sendmail** 的二进制文件及其配置文件，然后以守护进程的模式启动这个程序。用于您的 Linux 发行版本的 **sendmail** 软件包应该提供一个位于**/etc/init.d** 下正确的启动脚本。

有几种用户代理在它们向邮件系统提交一则用户的消息时，要显式地运行 **sendmail**（有时要带**-bm** 或者**-bs** 标志），而不是直接利用 SMTP 协议进行通信。在这种情况下，**sendmail** 使用配置文件 **submit.cf** 把消息放入**/var/spool/clientqueue** 这个排队用的目录。用**-Am** 或者**-Ac** 标志调用 **sendmail**，迫使消息分别发给 **mqueue** 或者 **clientqueue** 目录。



Red Hat 和 Fedora 有一个 **sendmail** 启动脚本（**/etc/init.d/sendmail**），它不会清理 **mqueue** 目录下的内容（如上面的例子所示）。不过，它确实能重构这个数据库。在采用 **sendmail** 8.12 及以后版本的情况下，可以将 LDAP 用于数据库映射，而且不必在改动之后重构映射。在 **/etc/sysconfig/sendmail** 中定义的参数决定了 **sendmail** 是否应该以守护进程模式启动，以及它应该以什么样的频度运行排队操作，所以您应该在那里针对您的站点设置 **sendmail** 启动时的行为。Red Hat 的发布以守护进程模式启动 **sendmail**，而且每 30 分钟运行一次排队操作。



SUSE 的启动脚本（**/etc/init.d/sendmail**）只是检查二进制和配置文件，然后用环境变量 SENDMAIL_ARGS 中定义的参数启动 **sendmail**，这些都是在文件**/etc/rc.config.d /sendmail.rc.config** 中设定的。SUSE 默认也以守护进程模式运行 **sendmail**，而且每 30 分钟运行一次排队操作。

如果您的发行版本没有包含一个 **sendmail** 启动脚本，那么应该在**/etc/init.d/sendmail** 中加进上面的 **sh** 代码片段。安装指导中有一个更漂亮的脚本，它会尝试清理干净以前被中断运行的排队操作。随您高兴是否采用。

历史上 **sendmail** 的支持文件已经在文件系统的各个可能之处——比如**/usr/lib**、**/etc/usr/ucblib** 和**/usr/share**——都呆过了。从 8.10 版的 **sendmail** 起，所有的文件（除了队列目录）都应该保存在**/etc/mail** 目录下[20]。让我们寄希望于厂商能够采纳这项暗示性的建议，把它们始终都放在一个地方。

[20] 所有的文件都保存在**/etc/mail** 下这个说法并不完全对。**sendmail.pid** 文件以及有时候的一些统计文件仍然留在了别的地方。

www.TopSage.com

18.6.3 在 Debian 和 Ubuntu 系统上安装 sendmail



您可以使用 **apt-get** 程序来安装 **sendmail**。它既安装 **sendmail** 和 **m4**，也要卸载 Exim。在 **apt-get** 已经下载并安装完 **sendmail** 软件包之后，它就要提示配置这个软件包。选择“是(yes)”，调用一个脚本，它会问您（大约 20 个）有关您所需要的 **sendmail** 配置的问题。每个问题都在中括号里给出了默认的答案，并且有几个特殊的地方给出了可行的配置文件。您需要回答不同于默认值的唯一一个问题就是“mail name”，默认的回答是不全的主机名（例如，lappie），但它需要的是完整的全名（例如，lappie.toadranch.com）。

如果您熟悉 **sendmail** 的各种选项和功能（它们将在本章的后面详细介绍），那么回答由脚本提出的问题就更有意义。这个配置脚本不考虑它默认包含的普通域文件的内容。因此，您可以不要一项功能，但最后它还是打开了（例如，`redirect` 功能）。

18.6.4 开关文件

Linux 系统都有一个称为“服务开关”的配置文件**/etc/nsswitch.con**，它会列举出可用于满足各种标准查询的方法，比如主机和用户的查找。服务开关还确定对给定类型的查询列出多种方法时，按照什么顺序尝试这些方法。服务开关在第 17 章中有更详细的讨论。

服务开关的操作通常对软件是透明的，但 **sendmail** 喜欢对自己的查找施加更细粒度的控制，于是忽略了系统上的开关文件，而代之以使用自己内部的一个开关配置（**/etc/mail/service.switch**）。

服务开关中影响邮件系统的两个字段是 `aliases` 和 `hosts`。对于 hosts 来说，主机服务可能的值是 `dns`、`nis`、`nisplus` 和 `files`。对于 `aliases` 来说，可能的值有 `files`、`nis`、`nisplus` 和 `ldap`。必须在 **sendmail** 二进制程序中编译进去对您所用所有机制的支持（除了 `files` 之外），才可以使用相应的服务。

sendmail 的内部服务开关包括：

```
aliases    files nisplus nis          # if compiled with nis/nis+
hosts      dns   nisplus nis files
```

18.6.5 运行模式

您可以在几种模式下运行 **sendmail**，用**-b** 标志进行选择。**-b** 代表“be”（是）或“become”（变成），它总是和别的标志一起使用，来确定 **sendmail** 将扮演的角色。表 18.4 列出了合法的值。

表 18.4 sendmail 主要模式的命令行标志

标 志	含 义
-bd	在守护进程模式下运行，监听端口 25 上的连接
-bD	在守护进程模式下运行，但在前台而不是后台[a]
-bh	查看最近的连接信息（等效于 **hoststat**）
-bH	清除过时的连接信息在硬盘上的副本（等效于 **purgestat**）
-bi	初始化散列的别名（等效于 **newaliases**）
-bm	作为邮寄程序（mailer）运行，以通常方式投递邮件（默认）
-bp	打印邮件队列（等效于 **mailq**）
-bP	通过共享内容打印队列中的项数（8.12 及以后版本）

www.TopSage.com

续表

标　志	含　义
-bs	进入 SMTP 服务程序模式（在标准输入上，不是在端口 25 上）
-bt	进入地址测试模式
-bv	只验证邮件地址，而不发送邮件

a. 这种模式用于调试，以便您可以看到错误和调试消息。

如果传入的邮件预计是从 Internet 上来的，那么要在守护进程模式（**-bd**）下运行 **sendmail**。在这种模式下，**sendmail** 将监听网络端口 25 并等待任务[21]。在守护进程模式下运行 **sendmail** 时，通常也会指定**-q** 标志，它设置的是 **sendmail** 处理邮件队列的时间间隔。例如，**-q30m** 或**-q1h** 将每 30 分钟或每小时运行一次排队操作。

sendmail 正常情况下会尝试立即投递消息，把它们暂时保存在队列中只是为了保证可靠性。但是如果您的主机太忙，或者无法到达目的地的机器，**sendmail** 就会把消息排入队列，稍后再发送它。**sendmail** 每次处理队列时会派生出子进程，所以不要把队列处理的时间间隔设置得太短了。RFC1123 推荐在两次运行之间至少间隔 30 分钟。**sendmail** 会进行上锁，所以多个队列同时运行也是安全的。

sendmail 8.12 已经增加了一种新功能，帮助处理大型的邮递列表和大型的队列：用信封划分队列组。这种功能详细介绍在 18.13.2 节。

sendmail 只在启动的时候读取它的配置文件 **sendmail.cf**。因此，在您改动过配置文件以后，必须杀死它再重新启动，或者向它发送一个 HUP 信号。**sendmail** 创建一个 **sendmail.pid** 文件，其中包含有它的进程 ID 和启动它的命令。您应该用绝对路径来启动它，因为在接收到 HUP 信号的时候，它会再次执行这个命令。**sendmail.pid** 文件能够让进程以下面的命令发 HUP 信号给 sendmail：

```
# kill -HUP `head -1 sendmail.pid`
```

PID 文件的位置以前是一个编译时刻的参数，但是它现在可以用**.mc** 配置文件里的 confPID_FILE 选项来设置。

```
define(confPID_FILE, '/var/run/sendmail.pid')
```

默认值与 OS 有关，但通常不是**/var/run/sendmail.pid** 就是**/etc/mail/sendmail.pid**。Red Hat、Fedora、Ubuntu 和 Debian 使用**/var/run/sendmail.pid**，而 SUSE 仍然把它保存在**/var/run/sendmail** 目录下。

18.6.6 邮件队列

当机器太忙碌以致不能立即投递邮件，或者当目标机器访问不到的时候，邮件消息就会被保存在队列目录中。**sendmail** 既能作为一个邮件提交代理，监听 587 端口，也能充当它通常的角色，作为一个守护进程，监听 25 端口并使用**/var/spool/mqueue** 这个目录作为队列目录。有些用户代理（例如，**mh** 和**/bin/mail**）使用端口 587，这是邮件提交代理的端口，而其他用户代理（Eudora、Outlook 等）直接使用 SMTP 协议和运行在 25 端口上的 **sendmail** 进行通信。从 8.12 版开始，邮件提交程序使用队列目录**/var/spool/clientmqueue** 和配置文件 **submit.cf** 把新消息注入到邮件系统中。所有消息来的时候都暂时进入这个队列。

sendmail 允许您有多个邮件队列，而且允许把这些队列的一个子集标为属于一个队列组（queue group）。例如，如果目录 **mqueue** 包括子目录 **q1**、**q2** 和 **q3**，并且您指定队列目录是**/var/spool/mqueue/q***，

[21] **sendmail** 所监听的端口由 DAEMON_OPTIONS 决定，默认值是端口 25。

www.TopSage.com

那么这 3 个队列都会被用到。**sendmail** 可以处理多个队列的能力提高了它在高负荷下的性能[22]。如果一个站点正在运行一个大型的邮递列表，**sendmail** 就把信封收件人列表分成几个较小的列表，给他们指派不同的队列组。这个技巧能够大大提高性能，因为可以并行处理多个较小的收件人列表。

队列组（queue group）是在 8.12 版 **sendmail** 中新出现的功能，它能对各种类型的消息做更细粒度的控制。在队列组上也可以设置任何与队列相关的参数，包括优先级（采用系统调用 **nice**）。邮件根据消息的第一个收件人的地址被提交给不同的队列组。默认的队列组称为 mqueue，它是自动定义的，不必做任何进一步的配置就能使用它。我们将在 18.13.2 节详细介绍队列组。

当消息被放入队列时，它被分段保存在好几个不同文件中。各个文件名有一个两字母的前缀，用来标识其片断，接着是从 **sendmail** 进程 ID 来的一个随机 ID。这个 ID 不是固定的，因为 **sendmail** 在不断派生，每个副本会获得一个新的进程 ID。表 18.5 列出了 6 个可能的片段。

表 18.5 邮件队列中的文件所用的前缀

前 缀	文 件 内 容	前 缀	文 件 内 容
qf	消息头和控制文件	**Tf**	表示上锁操作已经尝试过 32 次以上
df	消息主体	**Qf**	表示消息被弹回，不能返回
tf	正在更新 **qf** 文件时，它的一个临时版本	**xf**	邮寄程序（mailer）的出错消息的临时抄本文件

如果在队列目录下存在子目录 **qf**、**df** 或 **xf**，那些消息片段就会被放入适当的子目录中。**qf** 文件不仅包含消息头，还包含信封地址、消息无法投递时应该返回的日期、消息在队列中的优先级，以及消息在队列中的原因。每行由一个字母代码开头，这个字母会标识出这一行的其余部分。

每个排入队列的消息必须有 **qf** 和 **df** 文件。所有其他前缀由 **sendmail** 在尝试投递的过程中使用。当一台机器崩溃并重新启动时，**sendmail** 的启动命令序列应该从每个队列目录中删除 **tf**，**xf** 和 **Tf** 文件。负责邮件的系统管理员应该经常检查一下 **Qf** 文件，以防出现本地配置导致弹回的情况。

使用邮件队列很容易导致错误发生。例如，文件系统可能被填满（请避免把**/var/spool/mqueue** 和 **/var/spool/news** 放在同一个分区上），队列可能阻塞，孤立的邮件消息可能会困在队列中。

sendmail 有一个配置选项（confMIN_FREE_BLOCKS）可以帮助管理磁盘空间。当包含邮件队列的文件系统变得太满时，邮件将被“try again later（稍微再试）”错误拒绝，直到有更多可用的空间为止。这个选项会留下一小块溢出空间，这样在文件系统完全充满并使得一切都动弹不得之前，邮件就开始被拒绝了。

如果某个主邮件枢纽死机，它的 MX 备份站点就可能因为数以千计的消息而过载。**sendmail** 可能会派生出过多的副本而让机器死机。有几个选项对改善非常繁忙机器上的性能有帮助，我们将在 18.14.2 节集中介绍它们。为了处理一个被暂时阻塞的队列，须把阻塞移到一边，像平常一样继续处理新邮件，等一切平静下来再对阻塞队列运行 **sendmail** 的一个单独副本。有关 DNS MX 记录的更多信息请参见 15.7.6 节。例如，处理单个队列目录的过程类似于下面这样：

```
# kill `head -1 sendmail.pid`
# mv mqueue cloggedqueue          /* To another FS if necessary */
# mkdir mqueue                    /* Set owner/perms, too */
# chown root mqueue
# chmod 700 mqueue
# /usr/sbin/sendmail -bd -q1h &
```

在一切平息下来以后，用下面的标志运行 **sendmail**：

```
# /usr/sbin/sendmail -oQ/var/spool/cloggedqueue -q
```

[22] 只要目录不包含过多的文件，它们就是一种高效的存储机制。如果您的邮件服务器很繁忙，有大量过期的邮件列表，那么队列目录很容易大到不能高效处理的地步。

www.TopSage.com

这些标志把 **sendmail** 指向阻塞的队列目录，并指出 **sendmail** 应该立即处理它。重复这一命令直到队列为空。从 8.12 版开始，**sendmail** 使用了硬链接的方式，如果移动了一个队列，那么就会破坏它。处理拥塞队列的一种更好的办法是使用故障后备的主机的 MX 记录，参见 18.13 节了解详情。

队列在什么时刻会变得拥塞取决于站点和 **sendmail** 所正在运行的硬件。您的系统和 aol.com 的邮件枢纽（每天要处理数百万消息）采用的拥塞队列的定义是不一样的。参考 18.14 节了解有关如何测量流量水平的知识。

18.7 sendmail 的配置

在 8.12 版之前，**sendmail** 的操作是由**/etc/mail/sendmail.cf** 这一个配置文件控制的（以前可以在**/etc** 或者**/usr/lib** 下找到它）。我们把它简称为配置文件。版本 8.12 引入了第二个配置文件，它叫做 **submit.cf**（也在**/etc/mail** 目录下）。启动 **sendmail** 所采用的标志决定了它将使用哪一个配置文件：如果有 **submit.cf** 文件，那么**-bm**、**-bs** 和**-bt** 就使用它，而其他所有模式都使用 **sendmail.cf** 文件。当然，还可以用命令行参数和配置文件选项改变配置文件的名字，但最好保持这两个名字不变。配置文件决定了 **sendmail** 的以下几个方面：

- 投递代理程序的选择；
- 地址重写的规则；
- 邮件信头的格式；
- 选项；
- 安全预防；
- 垃圾邮件抵抗力。

原始配置文件的格式是专为分析方便而设计的。这种着眼点导致它有点儿缺乏一种热情友好的形象。维护这个配置文件是和电子邮件有关的管理工作中最重要的，而且甚至难倒了一些有经验的系统管理员。

sendmail 的每个版本都使用一个配置文件，但是现代的版本通过使用 **m4** 宏，让配置过程变得更容易了，因为它隐藏了大部分下层的复杂性。可以说原始的配置文件位于汇编语言的层次，而 **m4** 配置更像是处于 Perl 的层次[23]。

当第一次引入 **m4** 宏时，曾经希望它能处理 80%~90%的情况。而实际上，覆盖的范围要高得多，可能接近 98%。在本书中，我们只讨论基于 **m4** 的“简装版配置”。只有当您要调试棘手问题，或者以一种奇怪的方式扩展邮件站点时，才需要钻研底层的配置文件。

最关键的 2 份文档是：Eric Allman 发表的论文 *Sendmail Installation and Operations Guide*（位于 **sendmail** 软件发布的 **doc/op** 目录下），以及 README 文件（位于目录下）。我们经常把讲解安装的论文叫做“the installation guide（安装指南）”，而把 README 文件叫做 **cf/README**。

18.7.1 使用 m4 预处理器

我们先描述 **m4** 的一些功能，然后演示如何从一个 **m4** 主控文件构建一个配置文件，最后描述 **sendmail** 发布所带的预装 **m4** 宏中比较重要的部分。

m4 原打算用作程序设计语言的前端，让用户编写可读性更好（或者也许是更隐晦）的程序。但

[23] **sendmail** 的配置语言是“完全图灵机”，这意味着它可用于编写任何可能的计算机程序。和原始配置文件已经打过交道的读者将会认识到这是一个多么可怕的概念……

www.TopSage.com

TopSage.com

现在 **m4** 很强大，在许多输入转换的场合之下使用一点儿问题都没有，而且很适合处理 **sendmail** 的配置文件。

m4 宏的形式是：

```
name(arg1, arg2, ..., argn)
```

在 *name* 和左括号之间应该没有空格。作为参数的字符串要用左和右单引号引起来。**m4** 采用的引号习惯不同于您可能用过的其他那些语言，因为左引号和右引号是不同的字符[24]。引号也可以嵌套。有了今天的编译器构建工具，句法严格而奇异的 **m4** 能生存下来真是个奇迹。

m4 有一些内置宏，用户还可以定义他们自己的宏。表 18.6 列出了在 **sendmail** 的配置中最常见的内置宏。

表 18.6 sendmail 常用的 m4 宏

宏	功 能
define	定义一个名为 arg1，值为 arg2 的宏
undefine	丢弃一个以前定义的名为 arg1 的宏
include	包含（插入）名为 arg1 的文件
dnl	丢弃直到下一个换行为止的字符（包括那个换行符）
divert	管理输出流

一些站点在每一行的结尾添加一个 dnl 宏，以使转换后的**.cf** 文件整齐。没有 dnl，**m4** 会给配置文件添加额外的空行。这些空行不影响 **sendmail** 的表现，但是它们使得配置文件很难读。在我们的例子中省略了 dnl。其他站点在想用作注释行的开头处使用 dnl。

m4 实际上并不理会文件中的注释。像下面这样的注释：

```
# And then define the ...
```

不会完成您所期望的任务，因为 define 是一个 **m4** 关键字，它会被展开。请改用 m4 的 dnl 关键字（代表“delete to newline[删除至换行]”）。例如，

```
dnl # And then define the ...
```

会起作用。必须在 dnl 和注释本身之间包含一个空格，好让它被当作是一条 **m4** 命令。

18.7.2 sendmail 的配置片段

sendmail 软件发布包括一个 **cf** 子目录，其中包含 **m4** 配置所必需的所有部分：一个 **README** 文件和表 18.7 中列出的几个子目录。

表 18.7 配置子目录

目 录	内 容
cf	.mc（master configuration，主控配置）文件的样本
domain	在 Berkeley 的各种域上采用的 **m4** 文件样本
feature	实现各种功能的片段
hack	尚未定论的值或者实现的特殊功能

[24] 实际上可以用 changequote 宏改变引号字符，但是最好不要在 **sendmail** 中彻底改变 **m4** 这个句法，因为各种不同的宏都假定采用了这样的引号字符。

www.TopSage.com

续表

目　录	内　容
m4	基本配置文件和其他核心文件
ostype	与操作系统有关的文件的位置和特殊之处
mailer	描述常见邮寄程序（mailer）（投递代理）的 **m4** 文件
sh	**m4** 所用的 shell 脚本

cf/cf 目录包含**.mc** 文件的例子。实际上，它包含的例子多到了会把您给搞糊涂的地步。我们建议您把 **cf** 目录改为 **cf.examples**，并为自己的本地**.mc** 文件创建一个新的 **cf** 目录。如果您这样做了，请把 **Makefile** 和 **Build** 脚本复制到新的目录下，以便 **README** 文件中的指导仍然有效。最好把所有的**.mc** 配置文件也复制到一个集中的位置，而不是把它们留在 **sendmail** 软件发布中。如果您试图从一个**.mc** 文件建立**.cf** 文件，而且不在软件发布的层次结构中，那么将不得不对 **Build** 脚本的相对路径名做些修改。

18.7.3 从.mc 样板文件构建配置文件

在深入有关各种配置宏、功能和选项的长篇累牍的详细资料之前，我们将前后颠倒，创建一个“不加修饰”的配置来举例说明这一过程。我们的例子用于一个叶节点 foo.com，而主配置文件名为 **foo.mc**。

我们将把 **foo.mc** 放在我们新建的 **cf** 目录下。（通过 **m4**）转换后的配置文件将是同一目录下的 **foo.cf**，最后将把它作为 **sendmail.cf** 安装在**/etc** 或者**/etc/mail** 下。**/etc/mail** 是 **sendmail** 配置文件的标准位置，但是有许多发行版本用了**/etc**。

有些样板文件应该每个新的**.mc** 文件中都加：

```
divert(-1)
#### basic .mc file for foo.com
divert(0)
VERSIONID(`$Id$')
```

如果希望在文件的开头加上注释，第一行必须是一个 `divert` 语句，它将丢弃 **m4** 输出流中所有似是而非的无用信息，而且不需要 `dnl` 就可以有#风格的注释。接下来是注释，然后是另一个 `divert`。

用 `VERSIONID` 行（在这里是 RCS 格式）结束这个样板文件。这在下一个小节中有详细说明。

在许多情况下，指定一个 `OSTYPE`（参见 18.8.2 节），引进与操作系统相关的路径或参数，再指定一组 MAILER（参见 18.8.4 节），配置就完成了。

```
OSTYPE(`linux')
define(`confCOPY_ERRORS_TO', `postmaster')
MAILER(`local')
MAILER(`smtp')
```

在这里，我们还设置了一个选项（`confCOPY_ERRORS_TO`），它会把任何被弹回邮件的信头副本发送给本地 postmaster。当本地站点出现问题时，这项声明就能让 postmaster 介入。

要构建真正的配置文件，只需复制到新 **cf** 目录下的 **Build** 命令即可。

```
# ./Build foo.cf
```

或者

```
# make foo.cf
```

www.TopSage.com

最后，把 **foo.cf** 安装到适当的位置——通常在**/etc/mail/sendmail.cf**，但是 Red Hat 和 SUSE 都把它藏到了**/etc/sendmail.cf**。

较大的站点可以创建一个单独的 **m4** 文件，把站点范围的默认值存放在 **cf/domain** 目录中，这样单个主机可以包括这个文件的内容。不是每台主机都需要一个单独的配置文件，但是每组相似的主机（相同的体系结构和相同的角色：服务器、客户机等）可能需要其自己的配置。

18.7.4 改变 sendmail 的配置

您经常会发现自己现有的 **sendmail** 配置几乎全部都对，但您只想尝试一下一项新的功能，加入一条新的垃圾邮件规则，或者做一次简单的改动。要做到这一点，您就要：

- 编辑**.mc** 文件，输入您的改动；
- 用配置目录下的 **Build** 脚本重新构建配置文件；
- 把得到的 **cf** 文件安装为正确目录下的 **sendmail.cf**；
- 向 **sendmail** 发送一个 HUP 信号，让它重新读取配置文件[25]。

哪怕采用的是 **sendmail** 很容易的最新配置系统，您仍然必须针对您的站点做出几项配置决定。在您阅读下面介绍的功能的同时，考虑一下怎样才能把它们融入您的站点结构中。小规模的站点或许只需要一个枢纽结点和几个叶子结点，因而只需要两个配置文件的版本。较大规模的站点可能需要几个用于传入和传出邮件的邮件枢纽主机，或许还有一台单独的 POP/IMAP 服务器。

无论您的站点有多复杂，也无论它要面对怎样的外部世界（例如，完全暴露在外、在防火墙之后、或者在一个 VPN 上），很可能 **cf** 目录已经包含了一些现成的合适配置片段，正等着对它进行定制，然后投入工作呢。

18.8 sendmail 基本配置原语

sendmail 配置命令是区分大小写的。根据习惯，预定义宏的名称全部是大写（例如 `OSTYPE`），**m4** 命令全部是小写（例如 `define`），可配置的变量名用小写的 `conf` 打头，以全部大写的变量名结尾（例如 `confCOPY_ERRORS_TO`）。宏通常指向一个叫做*../macroname/arg1.m4* 的文件。例如，宏 `OSTYPE('linux')` 使得**../ostype/linux.m4** 被包括进来。

在本节中我们只涉及基本的配置命令，把高级命令留到后面讨论。

18.8.1 VERSIONID 宏

您应该用 CVS 或者 RCS 维护配置文件，这不仅是为了能在必要时返回到早先的配置版本，也是为了能够标出构成配置文件的 **m4** 文件的版本。请使用 `VERSIONID` 宏来自动嵌入版本信息。句法是：

```
VERSIONID('$Id$')
```

实际的版本信息将在您检入文件时由 RCS 填入。它将作为注释出现在最终的 **sendmail.cf** 文件中。如果您忘记把 **sendmail** 发布版本放在了哪个位置，这项信息对您会有帮助。文件的位置常常是由可用的磁盘空间而不是由文件系统的设计思路决定的。

[25] 使用 **kill** 命令来完成。有了 **sendmail.pid** 文件会更容易得知 **sendmail** 的进程 ID。遗憾的是，在不同的发行版本之间，它的位置并不固定（试试看**/var/run/sendmail.pid**）参考 18.6.6 节了解如何去做。

www.TopSage.com

18.8.2 OSTYPE 宏

ostype 目录下的文件用其包含的默认值所针对的操作系统来命名。一个 OSTYPE 文件封装了各种特定于厂商的信息，比如与邮件相关的文件的期望位置、sendmail 所需命令的位置、邮寄程序的标志等。

根据约定，特定于操作系统的信息由 OSTYPE 宏插入配置文件[26]。每个配置文件在靠近开头的地方都必须包括一个 OSTYPE 宏，一般是紧跟在 VERSIONID 之后。

OSTYPE 文件主要是通过定义其他 **m4** 变量来完成工作。例如：

```
define('ALIAS_FILE', '/etc/aliases')
```

指定系统范围的别名文件的位置。如果愿意的话，您可以稍后在**.mc** 文件中覆盖 OS 的默认值，但是请不要修改软件发布中自带的 OSTYPE 文件，除非它真的错了，此时还应该提交一份错误报告给 sendmail-bugs@sendmail.org。有些网点想让别名文件跨平台保持一致的位置，于是在他们的 DOMAIN 文件中重新定义了它的位置。

cf 目录中的 **README** 文件列出了在 OSTYPE 文件能够定义的所有变量。表 18.8 一方面举出了一些重要的变量，另外还有几个您可能想用来配置减少垃圾邮件的变量（但是默认没有定义）。默认值是您的 OSTYPE 文件没有指定其他值的时候您所得到的值。

表 18.8 在 OSTYPE 文件中设置的一些变量的默认值

变 量	默 认 值	变 量	默 认 值
ALIAS_FILE	**/etc/mail/aliases**	LOCAL_SHELL_PATH	**/bin/sh**
HELP_FILE	**/etc/mail/helpfile**	LOCAL_MAILER_MAX	未定义
STATUS_FILE	**/etc/mail/statistics**	LOCAL_MAILER_MAXMSGS	未定义
QUEUE_DIR	**/var/spool/mqueue**	SMTP_MAILER_MAX	未定义
MSP_QUEUE_DIR	**/var/spool/clientmqueue**	SMTP_MAILER_MAXMSGS	未定义
LOCAL_MAILER_PATH	**/bin/mail**		

如果在一个新的 OS 版本或体系结构上安装 **sendmail**，一定要创建一个相应的 OSTYPE 文件并把它提供给 sendmail.org，以便可以在 sendmail 的下一个版本中把它包括进来。只要以那些已存在的文件为模型建立新文件，然后对照 **cf/README** 文件中默认值的表检查它即可。如果新系统中某个变量的值与默认值相同，就不需要再为它包含一项（但是万一默认值有变化，这样做对保护您自己也没坏处）。

表 18.9 显示了我们举例的平台所用的 OSTYPE 文件。

表 18.9 Linux 系统的 OSTYPE 文件

系 统	文 件	目 录	用 法
Red Hat	**linux.m4**	**/usr/share/sendmail-cf**	OSTYPE('linux')
Fedora	**linux.m4**	**/usr/share/sendmail-cf**	OSTYPE('linux')
SUSE	**suse-linux.m4**	**/usr/share/sendmail**	OSTYPE('suse-linux')
Debian	**debian.m4**	**/usr/share/sendmail/sendmail.cf**	OSTYPE('debian')
Ubunutu	**linux.m4**	**/usr/share/sendmail**	OSTYPE('linux')

[26] 那么 OSTYPE 宏本身是在哪里定义的呢？在 **cf/m4** 目录下的一个文件中，当您运行 **Build** 脚本时，它会神奇地附加到配置文件的开头。

www.TopSage.com



SUSE把**sendmail**的软件发布放在**/usr/shsare/sendmail**里。OSTYPE文件**suse-linux.m4**就在那里的ostype目录下，它不是sendmail.org发布的**sendmail**软件包的一部分。那个文件很长（超过80行），包含了各种FEATURE，以及其他一些宏，它们通常可以在一个站点的主控配置文件（**.mc**文件）而不是OSTYPE文件里找到。这种做法把真实的配置向系统管理员藏了起来——或许混起来有好处，但我们建议在实际中不要这样做。



Debian把配置文件藏在了**/usr/share/sendmail/sendmail.cf/**下。**sendmail.cf**目录（Debian的人选的这个名字太容易让人搞混了）对应于**sendmail**软件发布中的**cf**目录，它包含您需要的所有文件，包括一个针对于Debian的OSTYPE文件**ostype/debian.m4**。这个OSTYPE文件有50行长，大多由路径名和注释所组成，它也应该是那样。许多内容和sendmail.org的当前默认内容是一样，所以不需要再明确地说明一遍。不过，重新说明一遍有助于防止Debian出现因为默认值的变化而造成的不一致性或者错误。只有Debian和**sendmail**的OSTYPE部分中不一样的内容包括在**sendmail**软件发布所带的一般DOMAIN文件里。在实际的**.mc**文件中，DOMAIN语句应该在前面出现，而不是隐藏在OSTYPE文件里。

18.8.3 DOMAIN宏

DOMAIN指令允许在一个地方（**cf/domain/***filename*.**m4**）指定全站的一般信息，然后在每台主机的单个配置文件中这样引用它：

```
DOMAIN('filename')
```

通常选择一个能描述站点特色的文件名。例如，我们给计算机科学系用的文件名为**cs.m4**，在**.mc**文件中表示为：

```
DOMAIN('cs')
```

和OSTYPE一样，DOMAIN其实只是一种包含的好方法。但它使得配置文件的结构更为清晰，并为将来的变化埋下了一个伏笔。它最大的用处是从保存在一个地方的**.mc**文件集中并构建网点的所有**.cf**文件。

小网点通常不需要DOMAIN文件，但是大些的网点常常用它们引用中转机器、站点级的伪装或保密选项，以及邮寄程序、虚拟域和垃圾邮件数据库的表。

软件发布所带的一般DOMAIN文件给出了通常放在全站范围DOMAIN文件中的那些类型的项。它的内容（删除了注释和dnl）如下所示：

```
VERSIONID(`$Id: generic.m4,v 8.15 1999/04/04 00:51:09 ca Exp $')
define(`confFORWARD_PATH', `$z/.forward.$w+$h:$z/.forward+$h:
    $z/.forward.$w:$z/.forward')
define(`confMAX_HEADERS_LENGTH', `32768')
FEATURE(`redirect')
FEATURE(`use_cw_file')
EXPOSED_USER(`root')
```

这个文件设置了用户转发文件的路径，限制了信头的长度[27]，包括进来用于已经离开单位的用户的重定向功能，以及打开了use_cw_file功能，以处理等价的主机名。如果您的.mc文件包括

[27] 黑客用非常非常长的信头作为对老版本**sendmail**的一种拒绝服务攻击手段。这行就是以防您仍然在运行这样的一种易受攻击的版本（8.9.3版之前）。

www.TopSage.com

大家网 TopSage.com

masquerading（伪装功能），那么 root 用户就不能被伪装。本章后面的内容将会详细介绍这些结构里的每一种。

18.8.4 MAILER 宏

您想要启用的每个投递代理都必须包括一个 MAILER 宏。在 **sendmail** 软件发布的 **cf/mailers** 目录下，可以找到一个被支持的邮寄程序的完整列表。当前，可选用的邮寄程序有 local、smtp、fax、usenet、procmail、qpage、cyrus、pop、phquery 和 uucp。一般说来，您至少需要：

```
MAILER('local')
MAILER('smtp')
```

第一行包括 local 和 prog 邮寄程序。第二行包括 smtp、esmtp、dsmtp、smtp8 和 relay。从 8.7 版[28]的 sendmail 开始，local 中加入了对 *user+details@site.domain* 这样的电子邮件地址的支持。*user* 定义了消息应该投递到的邮箱，而 *details* 提供了一个额外的参数，像 **procmail** 这样的本地邮件程序就能用这个参数来对传入的邮件进行分类。

如果您打算调整任何与邮寄程序相关的宏（比如 USENET_MAILER_ARGS 或 FAX_MAILER_PATH），要确保设置这些参数的配置行都位于调用该邮寄程序本身的行之前，否则只会使用旧的值。因此，通常把 MAILER 声明放在配置文件的结尾。

邮寄程序 pop 衔接 **spop** 程序，后者是 **mh** 邮件处理程序软件包的一部分，它实现了 RFC1460 中定义的 POP 协议（Post Office Protocol，邮局协议）。需要访问 UNIX 主机上的邮件的 PC 和 Mac 机器要用到它。邮寄程序 cyrus 用来配合 CMU 的 IMAP 服务程序使用，它有两种类型：cyrus 用于把邮件投递到用户的邮箱，cyrusbb 用于把邮件投递到一个中央布告栏。邮寄程序 cyrus 也能理解 user+details 式的句法，它的 MAILER 说明必须出现在 local 邮寄程序的说明之后。

邮寄程序 fax 把 Sam Leffler 的 HylaFAX 软件包集成到了邮件系统中。SUSE 包含的这个邮寄程序是**/usr/bin/faxmail**；Red Hat 和 Debian 默认不带 HylaFAX。向 *user@destination.fax* 发送邮件会把消息主体当作一个传真文档进行发送。*destination* 一般是一个电话号码。为了能将符号名（而不只是电话号码）作为 *destination*，要使用一个由关键字索引的数据库文件。从 www.hylafax.org 可以得到 HylaFAX。

您必须在**/usr/local/bin** 中安装 HylaFAX 软件发布中的一个脚本，才能把 HylaFAX 和 **sendmail** 关联到一起。您还可能修改宏 FAX_MAILER_PATH 的值。要把传入的传真从缓冲区投递到用户的邮箱，仍然需要人工介入。您可以把传真文档转换为 PostScript 文件（用 HylaFAX），并用 GNU 软件包 **ghostscript** 来查看它们。从 www.gnu.org 可以获得 **ghostscript**。

邮寄程序 qpage 衔接 QuickPage 软件，把电子邮件投递到您的呼机。有关 QuickPage 的更多信息请参见 www.qpage.org。

上述的宏 VERSIONID、OSTYPE、DOMAIN 和 MAILER 都是建立一个基本的 ***hostname*.mc** 文件所需要的。

18.9 sendmail 高级配置原语

在接下来的这几小节中，我们将介绍更多的宏，以及用来修改 **sendmail** 默认行为的一些最常见的 FEATURE 宏。我们还将讨论 **sendmail** 配置过程中的一些策略问题：安全、保密、垃圾邮件、以及使

[28] user+details 的语法源于卡内基梅隆大学（Carnegie Mellon University），他们用它配合本地工具一起路由和分类邮件。

www.TopSage.com

用伪装和虚拟域来隐藏信息的技术。

18.9.1 FEATURE 宏

利用 FEATURE 宏可以把来自 **feature** 目录的 m4 文件包括进来，从而启用几个常见的选项。在下面的讨论中，我们把 FEATURE 和 **sendmail** 的其他一些宏混在一起介绍，因为它们偶尔会纠缠在一起。当 **sendmail** 中首次加入 **m4** 配置的时候，对 FEATURE 宏的介绍就变成了我们这一章的很大一部分。现在，它又添加了那么多的功能，FEATURE 宏几乎需要它自己的专门一章来介绍了。

FEATURE 宏的句法是：

```
FEATURE(keyword, arg, arg, ...)
```

其中 *keyword* 对应于 **cf/feature** 目录下的文件 *keyword*.**m4**，而这几个 *arg* 会传递给它。确定功能的列表请参考这一目录本身的内容或 **cf/README** 文件。在下面会介绍一些常用功能。

18.9.2 use_cw_file 功能

sendmail 的内部类 w（因此名为 **cw**，代表 class name）包含这台主机要为之接受和投递邮件的所有本地主机的名称。一台客户机可能会把它的 hostname、它的 nickname 和 localhost 都包括到这个类里面。如果正在配置的主机是您的邮件枢纽主机，那么这个类 w 中还应该包括要为之接受电子邮件的所有本地主机和虚拟域。

use_cw_file 功能从文件**/etc/mail/local-host-names** 中定义类 w（过去把它称为 **sendmail.cw**）。可以用 confCW_FILE 选项配置确切的文件名，这将在后面讨论。没有这项功能的话，**sendmail** 只接受发给它正在运行的那台机器的邮件。传入邮件的服务器必须在 **local-host-names** 文件中列出它为之处理邮件的所有主机和域名。因为 **sendmail** 只在启动时读取.cw 文件，所以如果修改了.cw 文件，就必须向 **sendmail** 发送一个 HUP 信号才能让更改生效。

```
FEATURE('use_cw_file')
```

调用此功能并使用 **local-host-names** 作为数据来源的文件，下面是一个 **local-host-names** 文件的例子：

```
# local-host-names - include all aliases for your machine here.
toadranch.com
coyote.toadranch.com
big-tr.com
yoherb.com
herbmorreale.com
appliedtrust.com
applied-trust.com
atrust.com
```

在这个例子里，这些项都是本地托管的虚拟域。

18.9.3 redirect 功能

当有人离开单位时，您通常会转发他们的邮件，或者让发给他们的邮件带着一条错误消息弹回给发件人。redirect 功能支持一种更文雅的弹回邮件的方法。如果 Joe Smith 从 oldsite.edu 毕业并来到 newsite.com，那么可这样启用 redirect：

```
FEATURE('redirect')
```

www.TopSage.com

并把下一行：

```
smithj:  joe@newsite.com.REDIRECT
```

添加到别名文件中，这会使得发给 smithj 的邮件被返回给发件人，并带有一则错误消息，建议发件人尝试改用地址 joe@newsite.com。这则消息本身并不会被自动转发。

18.9.4 always_add_domain 功能

这项功能会让 **sendmail** 把本地主机名添加到不完整的本地目的地址上。例如，假设从 lynda@cs.colorado.edu 发邮件给本地用户 barb 和 evi。如果没有 always_add_domain，那么 barb 收到的邮件信头中就只把登录名作为发件人和收件人地址显示出来。如果打开了 always_add_domain，那么在消息离开 lynda 的机器时，所有的地址都会变成完整的全名。

当您在机器之间共享缓冲区目录，而这些机器并不共享一个别名文件或不具有相同的 **passwd** 文件时（顺便提一句，这样的共享可能是不应该的），也适合使用 always_add_domain。在给某个并非众所周知的别名或用户发邮件时，在源机器上就会让它带有完整的地址，因此就可以回复这封邮件了。

这项功能的另一个优点是，不完整的名字经常被当作垃圾邮件而被拒收。我们建议您一定使用这项功能（除非是您在发垃圾邮件！）。

如果您正在使用 MASQUERADE_AS（参考 18.9.11 节），always_add_domain 会添加您所伪装成的主机名，而不是本地主机名。如果本地主机上的别名文件或 **passwd** 文件不是 MASQUERADE_AS 主机上对应文件的子集，这种约定会导致出问题。

18.9.5 nocanonify 功能

sendmail 一般要验证一个地址的域名部分是否是一个完整的域名，而不是一个 DNS CNAME。如果不是这样的话，**sendmail** 会重写地址。这个过程称为规范化（canonification），通常要对主机名执行一次 DNS 查询来完成这个过程。nocanonify 功能将不做重写，并推迟发送消息所必需的 DNS 查找。例如，某个网点有一台主邮件枢纽，而客户机器通过主枢纽来转发他们的所有邮件，则客户可能会用：

```
FEATURE('nocanonify')
```

来避免在本地执行 DNS 查找。在一则消息在本地站点内从一台机器传送到另一台机器的时候，**sendmail** 并不跟踪确定是否完成了 DNS 查找——它不能这样做。nocanonify 功能让您可以控制这些 DNS 查找的时机。

nocanonify 还可以用于 MSA/MTA 方案中，这种方案可在非常大的邮件站点中使用。在这种情况下，由 MSA 完成所有的 DNS 查找，运行 MTA 的主枢纽机指定 nocanonify。

您有时候想要避免开销可能会比较大的 DNS 查找操作，但是您又愿意对本地域做查找。您可以用 CANONIFY_DOMAIN 或者 CANONIFY_DOMAIN_FILE 宏，从 nocanonify 说明中扣除掉特定的一些域，这两个宏分别可以接收一个域的列表，或者一个文件作为参数。例如，下面两行：

```
FEATURE('nocanonify')
CANONIFY_DOMAIN('cs.colorado.edu  cs')
```

会推迟除了 *user*@cs.colorado.edu 或者 *user*@cs 形式的地址之外的 DNS 查找操作。这几个特别的宏是在 8.12 版 **sendmail** 中首次引入的。

www.TopSage.com

18.9.6 表和数据库

sendmail 有几个 FEATURE 使用一种称为“表（table）”的结构来推断出邮件应该传递到哪里。一张表就是一个路由、别名、策略或其他信息的文本文件，这些信息从外部用 **makemap** 命令转换为某种数据库格式，然后作为一个内部数据库用于 **sendmail** 的各种查找操作。虽然这些数据通常一开始是一个文本文件，但是并不要求一定这样，**sendmail** 表中的数据可以来自 DNS、NIS、LDAP 或者其他来源。使用集中式的 IMAP 或 POP 服务器可以缓解 **sendmail** 找出用户是谁的麻烦，而且还省掉了下面讨论的一些表。后面的表 18.10 总结了可以用的表。

有两种数据库得到支持：**dbm/ndbm** 库是大多数版本的 Linux 的标准；而 Berkeley DB 是一种更具可扩展性的库，它支持多种存储方案。数据库的选择必须在编译时指定。如果可以安装 DB 的话，我们推荐使用它，它比 **dbm** 快而且创建的文件较小。DB 可以从 sleepycat.com 获得。

数据库映射类型有 3 种可用：

- dbm——使用扩展的散列算法（**dbm/ndbm**）；
- hash——使用标准的散列法方案（DB）；
- btree——使用 B-树数据结构（DB）。

对于 **sendmail** 中大多数要应用表的地方来说，数据库类型 hash——默认的——是最佳选择。使用 **makemap** 命令从文本文件中构建数据库文件，由您指定数据库的类型和输出文件的基准名。数据库的文本版本应该出现在 **makemap** 的输入中，例如：

```
# makemap hash /etc/mail/access < /etc/mail/access
```

乍看起来，这个命令似乎有错，会让输出文件被一个空的输出文件所覆盖。但是 **makemap** 添加了一个合适的后缀，所以实际的输出文件是**/etc/mail/access.db**，因此实际上没有冲突。每次改变了文本文件的内容之后，都必须用 **makemap** 命令重构数据库（但是不需要向 **sendmail** 发送 HUP 信号）。

在大多数情况下，数据库的关键字使用最长可能匹配的方式。和任何采用 hash 处理的数据结构一样，输入的文本文件中项的顺序并不重要。期待将数据库文件用作参数的 FEATURE 会把 hash 默认为数据库类型，把**/etc/mail/***tablename*.**db** 默认为数据库的文件名。为了覆盖这样的处理方式，要么可以给 **makemap** 命令和 FEATURE 指定期望的类型，要么可以通过给变量 DATABASE_MAP_TYPE 定义其他的值来重新设置默认值，例如：

```
define('DATABASE_MAP_TYPE', 'dbm')
```

为了使用新的 **access.db** 数据库，应该把下面的行添加到**.mc** 文件中：

```
FEATURE('access_db', 'hash /etc/mail/access')
```

因为这一行使用默认的类型和命名方案，所以可以只写为：

```
FEATURE('access_db')
```

您可以指定数据库的文件名，后缀（**.db**）可有可无，没有后缀更好。

每次修改文本文件后不要忘记用 **makemap** 重构数据库文件，否则改动不会生效。

我们在接下来的几小节介绍 mailertable、genericstable 和 virtusertable 这几个 FEATURE。在介绍垃圾邮件的一节里介绍 access_db。之所以根本不介绍 user_db，是因为人们已经在反对使用它，它最后会被删除。

从 8.12 版的 **sendmail** 开始，所有的映射和类都能把 LDAP 作为它们的数据来源，所以您可以让 **sendmail** 联络 LDAP 服务器来决定邮件的路由选择，以及信头的重写处理。只要把 LDAP 当作第二个

www.TopSage.com

参数即可：

```
FEATURE('access_db', 'LDAP')
```

这一行让 access_db 采用 **sendmail** 软件发布中的 **cf/sendmail.schema** 文件里所定义的默认 LDAP 方案。您还可以定义您自己的数据库方案，这要用到 FEATURE 指令的其他参数，参见 **cf/README** 文件了解详情。

18.9.7 mailertable 功能

mailertable 功能把寻址到特定主机或域的邮件通过一种特殊的邮寄程序重定向到替代的目的地。它在邮件从站点发出时起作用。mailertable 功能只检查地址中的主机部分，而不检查用户部分。信封地址不会被重写，因此邮件的地址仍然是同一个用户，但是是通过另一个邮寄程序发给另一台主机。mailertable 最初是为应付其他的邮件系统，比如 UUCP、DECnet 和 BITNET 而设计的，但现如今这些系统通常用来把邮件从一个网关主机重定向到一台内部服务器。

要使用一个 mailertable，可以在**.mc** 文件中包括下面这样一行：

```
FEATURE('mailertable')
```

在 mailertable 里的项采用的形式如下：

```
old_domain    mailer:destination
```

在左边关键字前面的点（.）是一个通配符，它代表该域中的任意主机。只有主机和域名允许作为 mailertable 的关键字，用户名则不允许是它的关键字。右边 *destination* 的值可以是一个域，一个 user@new_domain 子句，或者甚至可以为空（null），此时将不修改邮件信头。*mailer* 的值必须是在一个 MAILER 子句中定义过的一个邮寄程序的名称，参见 18.8.4 节。

举个例子，假如您使用 MS Exchange 作为域的内部邮件服务器，但是又不太愿意让它面对 Internet。您可以在 Internet 上放一台 Linux 机器，作为您的邮件网关，然后在对所有的邮件做病毒扫描或者您愿意做的任何处理之后，转发给 Exchange 服务器。下面是能够达到上述目的的 mailertable 项，假定 Exchange 服务器有如下所示的内部 IP 地址：

```
my-domain    esmtp:[192.168.1.245]
```

不过，就像我们会在 18.10.1 节看到的那样，这是一种中转（relaying）的形式。您还需要在 access 数据库中加上下面这行：

```
To:  my-domain  RELAY
```

允许转发所有给 my-domain 的任何用户的邮件。这样，就完成了上面的例子。

18.9.8 genericstable 功能

genericstable 功能（是指“generics table”，而不是“generic stable”）类似于给要发出的邮件起别名。例如，它可以把发给 trent@atrust.com 的邮件映射到 trent.hein@atrust:com。被重写的是信头而不是信封。邮件投递不受影响，只有回复会受影响。

用于映射主机名的机制有几种，但只有 genericstable 可以在映射关键字中同时包括用户名和主机名。在这一节中稍后讨论的 masquerade_envelope 和 allmasquerade 功能也可以应用到 genericstable 中的地址上。

www.TopSage.com

只有域在 generics 类才能用 genericstable。要在 generics 类加入一个域，既可以在 GENERICS_DOMAIN 宏中列出它，也可以把它放入由 GENERICS_DOMAIN_FILE 宏所指出的文件中。

例如，为了配合数据库的默认值一起使用 genericstable 功能，可以把：

```
GENERICS_DOMAIN_FILE('/etc/mail/local-host-names')
FEATURE('genericstable')
```

添加到.mc 配置文件中。在本例中，您为之接受邮件的任何主机都被包括进来了。启用 genericstable 功能会稍微降低 **sendmail** 的速度，因为要检查每个发件人的地址。

18.9.9 virtusertable 功能

虚拟用户表支持传入邮件的域别名。这项功能允许在一台机器上存在多个虚拟域，这在 Web 主机托管站点中很常见。

虚拟用户表的关键字字段要么包含一个电子邮件地址（*user@host.domain*），要么包含一个域的说明（*@domain*）。值字段是一个本地电子邮件地址或者一个外部电子邮件地址。如果关键字是一个域，那么值可以把 *user* 字段作为变量%1 传递，或者把邮件传递给其他用户。如果用户的说明采用 *user+details* 这样的形式，那么变量%2 将包含 *details*，而变量%3 将包含+*details*。随便使用您自己愿意采用的方式。

让我们来看一些例子（我们已经加上了注释）：

```
info@foo.com foo-info            # 送给一个本地用户
info@bar.com bar-info            # 另一个本地用户
joe@bar.com error:No such user # 返回一条出错信息
@baz.org jane@elsewhere.com      # 所有的邮件都给 jane
@zokni.org %1@elsewhere.com      # 给域名不同的同一用户
```

这些数据映射左边的所有主机关键字都必须在 **cw** 文件**/etc/mail/local-host-names**（或者 VirtHost 类）中存在；否则，**sendmail** 将尝试在 Internet 上寻找该主机并把邮件投递到那里。如果 DNS 把 **sendmail** 指回本服务器，您就会收到一则弹回的“local configuration error（本地配置错误）”消息。很遗憾，**sendmail** 不能分辨出这种情况下的错误消息应该是“virtusertable key not in cw file（virtusertable 关键字不在 cw 文件中）”。

实际上涉及的事件有下面几项。

- DNS MX 记录必须存在，以便首先把邮件传递到正确的主机。
- 然后，必须存在 cw 项，或者必须指定 VIRTUSER_DOMAIN，允许本地机器接受该邮件（或指定等价的 VIRTUSER_DOMAIN_FILE）。
- 最后，虚拟用户表必须告诉 **sendmail** 如何处理该邮件。

采用下面的这行调用这项功能：

```
FEATURE('virtusertable')
```

本章后面 18.10 节中的例子使用 virtusertable 来实现虚拟主机托管。

18.9.10 ldap_routing 功能

在别名、重写和篡改方面，我们最后介绍 LDAP，即 Lightweight Directory Access Protocol（轻量级目录访问协议）。对于传递电子邮件和接受虚拟域的邮件来说，LDAP（参见 18.5.2 节了解一般性的知识）可用来替代 virtusertable。它还可以管理别名、映射和类。到 8.12 版的时候，它在邮递列表上也能做得像模像样。

www.TopSage.com

要以这样的方式使用 LDAP，必须在配置文件中包括几条语句，并且必须编译构建 **sendmail** 把对 LDAP 的支持包括进来。在**.mc** 文件中需要下面几行：

```
define('confLDAP_DEFAULT_SPEC', '-h server -b searchbase')
FEATURE('ldap_routing')
LDAPROUTE_DOMAIN('my_domain')
```

来告诉 **sendmail**，您希望在传递寻址到指定域的传入邮件时，使用一个 LDAP 数据库。LDAP_DEFAULT_SPEC 选项标识出 LDAP 服务器和数据库搜索的基准名称。

在下面的例子中，搜索基准是 o=sendmai.com，c=US。如果在另一个自定义的端口（不是 389）上运行 LDAP，请把-p *ldap_port#*添加到 LDAP_DEFAULT_SPEC 中。

sendmail 要用到 LDAP 数据库中两个标记的值：

- mailLocalAddress 用于传入邮件中的地址；
- mailRoutingAddress 用于发送邮件的目的地的别名。

sendmail 还支持标记 mailHost，它如果存在，就把邮件传递给指定主机的 MX 记录，并把 mailRoutingAddress 作为收件人。

例如，LDAP 项（用于一个服务程序，配置该服务程序时用根的区别名 cn=root，o=sendmail.com，c=US）：

```
dn: uid=eric, o=sendmail.com, c=US
objectClass: inetLocalMailRecipient
mailLocalAddress: eric@sendmail.org
mailRoutingAddress: eric@eng.sendmail.com
```

这将使得发给 eric@sendmail.org 的邮件（其 DNS MX 记录使得邮件将被投递到 sendmail.com）被发到 eric@eng.sendmail.com。如果这一项还包含以下一行：

```
mailHost: mailserver.sendmail.com
```

那么发给 eirc@sendmail.org 的邮件地址将被改为 eric@eng.sendmail.com，在查找 MX 记录后，发给主机 mailserver.sendmail.com。

LDAP 数据库项支持通配符*@domain*，它把要发给指定域中任何人的邮件都重新确定发送路由（就像在 virtusertable 中所做的那样）。

在 8.12 及以后的版本中，以一种配置原语 LDAPROUTE_EQUIVALEN（或者 LDAPROUTE_EQUIVALENT_FILE）的形式加入了一点儿灵活性，它能让您定义采用 LDAP 重新路由的等价域名。例如，发送到 user@host1.mydomain 的邮件在正常情况下要在 LDAP 数据库中进行查询，然后作为@host1.mydomain 进行查询。包括下面：

```
LDAPROUTE_EQUIVALENT('host1.mydomain')
```

这行也会尝试关键字 user@mydomain 和@mydomain。这个功能可以在一个复杂的站点使用一个数据库来路由邮件。

ldap_routing 功能的另一个参数现在能够让您指定有关 LDAP 方案的更多细节，以此来使用和指定有一个+detail 部分的用户名的处理方式。截至 8.13 版，ldap_routing 的一个新参数（sendertoo）能拒绝从 LDAP 里不存在的发送者发来的 SMTP 邮件。同样，参考 **cf/README** 文件了解确切的细节。

18.9.11 伪装和 MASQUERADE_AS 宏

有了 MASQUERADE_AS 宏，您就可以指定一个身份，其他机器可隐藏在这个身份后面。所有邮件

www.TopSage.com

看起来都源自于这个被指定的机器或者域。发件人的地址被重写为 *user@masquerading-name*，代替了原来的 *user@original-host.domain*。当然，那些被伪装起来的地址必须是有效的，以便人们可以回复邮件。

user@houstuser 这项配置允许在一个网点上的所有用户使用一个通用的电子邮件地址。例如，如果 xor.com 上的所有主机都隐藏在域 xor.com 之后，那么来自 *user@host.xor.com* 的邮件上的邮戳表明好像它是来自 *user*@xor.com 一样，而且不会提到用户从主机发送邮件时的实际的主机名。代表 xor.com 的机器必须知道如何投递所有用户的邮件，甚至是在传入邮件服务器上不具有登录名的用户的邮件。自然，登录名在整个域之内必须是唯一的。

一些用户和地址（比如 root、postmaster、hostmaster、trouble、operations、Mailer-Daemon 等）应该不在这种做法的范围内。可以用 EXPOSED_USER 宏把它们明确地排除在外。例如，下面的配置序列：

```
MASQUERADE_AS('atrust.com')
EXPOSED_USER('root')
EXPOSED_USER('Mailer-Daemon')
```

将给邮件盖上它出自 *user*@atrust.com 的戳，除非它是由 root 或邮件系统发送的。在后面这些情况下，邮件将携带源主机的名称。

8.12 版的 sendmail 引入的一项功能可以让您从伪装功能中把本地域（或者发送到列为特殊的特定主机的邮件）排除在外。例如，对于一个既使用了没有注册的本地私用域名，又想对发往 Internet 的消息采用伪装的站点来说，这个功能可能会带来方便。

这个功能的语法为：

```
FEATURE('local_no_masquerade')
MASQUERADE_EXCEPTION('host.domain')
MASQUERADE_EXCEPTION_FILE('filename')
```

有几种对基本的 MASQUERADE_AS 宏进行扩展的方法，它们都是通过其他宏和 FEATURE 来实现的：

- MASQUERADE_DOMAIN 宏；
- MASQUERADE_DOMAIN_FILE 宏；
- MASQUERADE_EXCEPTION 宏；
- MASQUERADE_EXCEPTION_FILE 宏；
- limited_masquerade FEATURE；
- allmasquerade FEATURE；
- masquerade_envelope FEATURE；
- masquerade_entire_domain FEATURE。

我们推荐使用上面介绍的 MASQUERADE_AS 宏以及 allmasquerade 和 masquerade_ envelope 功能。limited_masquerade 功能可改变 MASQUERADE_DOMAIN 的行为，这对虚拟主机环境很有用。MASQUERADE_DOMAIN 允许您列出希望伪装的域，列表是从 w 类（这个类在典型情况下由 use_cw_file 功能定义）中预加载的，而且它列出了域中的主机。limited_masquerade 不用类 w 预先初始化这个列表。所有那些域都将隐藏在您所伪装成的域之后。

allmasquerade 功能把伪装扩展到消息的收件人（与仅仅伪装发件人相反），而 masquerade_ envelope 功能则把它扩展到信封以及信头地址[29]。有了这两项扩展，所有地址都能以一致的风格隐藏起

[29] 信头地址是出现在消息信头中的 To、From、Cc 和 Bcc 地址。信封地址是邮件实际上投递到的地址。信封地址最初是由用户代理程序从信头地址中构造的，但是它们要由 **sendmail** 单独进行处理。如果不保持信头地址和信封地址之间的区别，**sendmail** 的许多伪装和重定向功能就不可能实现。

www.TopSage.com

来。这个masquerade_entire_domain功能把伪装扩展到其他指定出的一系列域里的所有主机。

如果您希望使用其他伪装技术，可以在**cf/README**文件或在**sendmail**书中读读它们的功能。请仔细阅读，某些伪装原语可能隐藏得太多。

18.9.12 MAIL_HUB和SMART_HOST宏

通过重写信头和信封（可选），伪装使得所有邮件看起来都出自一个主机或者域。某些网点可能希望所有邮件真正地出自（或去往）单独的一台机器。把宏MAIL_HUB用于传入的邮件，把SMART_HOST用于发出的邮件，即可获得这种配置。

如果希望把所有传入的邮件传递到一个中央服务器来投递，请把MAIL_HUB的值设置为*mailer:host*，其中*mailer*是用来到达指定*host*的代理程序。如果不指定投递代理程序，就使用relay。例如：

```
define('MAIL_HUB', 'smtp:mailhub.cs.colorado.edu')
```

指定SMART_HOST会使得主机只投递本地邮件，而把外部邮件转给SMART_HOST。这项功能对位于防火墙后面不能直接使用DNS的机器很有用。其句法类似于MAIL_HUB的句法，默认的投递代理程序还是relay。例如：

```
define('SMART_HOST', 'smtp:mailhub.cs.colorado.edu')
```

在这些例子中，由同一台机器充当传入和发出邮件的服务器。更大的网点可能会把它们分到单独的机器中。SMART_HOST必须允许中转（relaying）功能，这样一来，客户机就能通过它发送邮件。mailertable项覆盖了SMART_HOST。

18.9.13 伪装和路由

所有这些功能和宏就绪并等待对您的电子邮件地址进行调整的时候，我们觉得根据它们是否修改了消息的信头、信封或投递是应用于传入消息还是传出的消息，是应用于发送方还是接收方的地址等，来尝试比较一下各种机制也许是有益的。如果本页面再宽上2~3倍，我们也许就能真正成功地举例说明各种结构之间的差别。

而现在，我们只能在表18.10里给您一点提示，您不得不在**sendmail**文档中查阅详细资料才能了解不同变化之间的所有细微差别。

表18.10中全部大写的项是**m4**宏，小写项是用FEATURE宏调用的功能的名称。缩排的项依赖于其上层的项，例如，除非MASQUERADE_AS宏已经打开，否则修改MASQUERADE_AS行为的功能将毫无用处。这张表把功能进行缩排以便反映出这种依赖关系。伪装功能会影响发出的邮件上的信头地址，以及一则消息是否可以回复；路由选择功能会影响邮件的实际投递过程。

表18.10 伪装和路由选择功能的比较

	结构	方向	影响[a]	哪一部分
伪装	MASQUERADE_AS	出	SH	host.domain
	allmasquerade	出	RH[b]	host.domain
	MASQUERADE_DOMAIN[_FILE]	出	SH	host.domain
	masquerade_entire_domain	出	SH	host.sub.domain
	limited_masquerade	出	SH	host.domain
	masquerade_envelope	出	SE[c]	host.domain
	genericstable	出	SH	user@host.domain

www.TopSage.com

续表

	结 构	方 向	影 响[a]	哪一部分
路由选择	mailertable	出	MAD	host.domain
	virtusertable	入	RD	user@host.domain
	ldap	入	RD	user@host.domain
	mailhub	入	RD	本地邮件
	smarthost	出	RD	远程邮件

a. S=发件人、R=收件人、D=投递、H=信头、E=信封、M=邮寄程序、A=地址。
b. 一旦用 allmasquerade 功能启动了重写收件人的处理，其他所有的伪装结构不仅要重写发件人，也要重写收件人。
c. 一旦用 masquerade_envelope 功能启动了信封重写的处理，那么其他所有的伪装结构不仅要重写信头，也要重写信封。

18.9.14 nullclient 功能

nullclient 功能用于永远不直接接收邮件，而且将其所有发出的邮件发送给中央服务器的主机。这样一台主机所用的**.mc** 文件只有两行。

```
OSTYPE('ostype')
FEATURE('nullclient', 'mail_server')
```

nullclient 功能将忽略许多其他功能。所有邮件都将毫无例外地交给 *mail_server* 处理[30]。请注意，如果用户正常情况下是在客户机上发邮件，而且没有为发出的邮件使用一个独立的服务器，则该服务器必须允许客户通过这个服务器进行中转（relay）。最近的 **sendmail** 版本在默认情况下关掉了中转功能。有关如何控制中转的详细资料请参考 18.10.1 节。nullclient 配置伪装成了 *mail_server*，那么您可能希望包括一条用于 root 的 EXPOSED_USER 子句。

使用 nullclient 功能的客户机必须拥有一条指向该服务器的相关 MX 记录。它也必须包括在服务器的 **cw** 文件中，该文件通常是**/etc/mail/local-host-names**。这些设置让服务器能够为那台客户机接受邮件。

使用 nullclient 配置的主机应该不能接受传入的邮件。如果它这样做了，它无论如何要把邮件转发给服务器。启动 **sendmail** 时不要使用**-bd** 标志，以使它不去监听端口 25 上的 SMTP 连接。不过，有些用户代理（MUA）试图通过端口 25 来提交邮件消息，所以如果您的 **sendmail** 没有监听端口 25，那么会有妨碍。不允许有传入邮件更好的办法是用**-bd** 标志运行 **sendmail**，但是用 DAEMON_OPTIONS 宏只监听 **loopback** 接口。无论哪一种方法，都应该在命令行上保留**-q30m** 标志，这样如果 *mail_server* 停机，客户就可以把要发出的邮件先放入队列，稍后再尝试把它发送给 *mail_server*。

nullclient 适用于拥有一台中央邮件机器的网点中的叶节点。在较大的网点中，请考虑中央机器上的邮件负荷。您可能想把传入和发出邮件的服务器分开，或采用某种层次结构的做法。



SUSE 带有一个 nullclient 的 **mc** 文件样本**/etc/mail/linux.nullclient.mc**。只要填上您的 *mail_server* 的名字，在它上面运行 **m4** 就能构造出 **sendmail.cf**，要做的就这么多。

18.9.15 local_lmtp 和 smrsh 功能

在默认情况下，本地邮寄程序使用**/bin/mail** 作为用户和文件的本地投递代理，使用**/bin/sh** 作为程

[30] 如果用这种方法配置一台客户机，那么用 **sendmail-bt** 测试这个配置，客户机会在本地投递本地邮件。原因是 nullclient 指令是稍后在原始配置文件中的规则集 5 里处理的。

www.TopSage.com

大家网 TopSage.com

序的投递代理。**sendmail** 现在提供了更好的选择，特别是对于程序的投递来说更是如此。两者都可以用 FEATURE 宏实现。

如果指定了 `local_lmtp` 功能，那么它的参数就是一个能够支持 LMTP 协议（Local Mail Transport Protocol，本地邮件传输协议，参见 RFC2033）的本地邮寄程序。默认向用户投递邮件的是 **sendmail** 软件发布带的 **mail.local** 程序。

类似地，`smrsh` 功能指定了用于向程序投递邮件的程序的路径。参见 18.12.3 节了解有关 **mail.local** 和 **smrsh** 的详细讨论。

18.9.16 local_procmail 功能

您可以通过启用 `local_procmail` 功能来使用 Stephen van den Berg 的 **procmail** 作为您的本地邮寄程序。它接受 3 个参数：**procmail** 二进制文件的路径、调用它所用的参数向量、邮寄程序的标志。默认值都比较合适，但是默认路径（**/usr/local/bin/procmail**）和大多数发行版本上 **procmail** 常见的位置（**/usr/bin**）有冲突。

procmail 能够为用户做到比纯粹的 **/bin/mail** 或者 **mail.locol** 更复杂的事情。除了把邮件投递到用户的信箱里之外，它还能把消息分类放入不同的文件夹、把它们保存在文件里、运行程序和过滤垃圾邮件。采用 `local_procmail` 功能大大抵消了 **smrsh**（在 18.12.3 节里介绍）在安全方面所做的增强。不过，如果您不需要限制您的用户所能运行的程序（也就是说，您信任所有的用户），那么 **procmail** 就会非常方便。**procmail** 并不随 **sendmail** 一起发布，如果您的厂商没有安装它，那么您可以从 www.procmail.org 得到它。

只要欺骗 **sendmail** 说您只是给了它一个 **procmail** 本地的副本，就可以配合这项功能使用别的邮件处理程序：

```
FEATURE('local_procmail', '/usr/local/bin/mymailer')
```

如果您使用 **procmail**，那么要检查 **/usr/bin/mailstat** 了解一些方便的 **procmail** 统计信息（不要和 **/usr/sbin/mailstats** 混淆了，后者给出了 **sendmail** 的统计信息）。在我们举例的所有发行版本中都有它，而且可以用它来总结 **procmail** 的日志文件。

18.9.17 LOCAL_*宏

如果您确实不怕麻烦，要编写一些新奇的新规则以便处理特殊的本地情况，请使用一组以 `LOCAL_` 打头的宏。本章后面的 18.10 节中有一些这种低级结构的例子。

18.9.18 配置选项

配置文件的选项和宏（原始配置语言中的 `O` 和 `D` 命令）可以用 **m4** 命令 `define` 设置。在 **cf/README** 文件中列出了作为 **m4** 变量访问的选项的完整列表及其默认值。这些默认值对于大多数站点来说都适用。

举几个例子：

```
define('confTO_QUEUERETURN', '7d')
define('confTO_QUEUEWARN', '4h')
define('confPRIVACY_FLAGS', 'noexpn')
```

QUEUERETURN 选项决定了一则消息如果不能投递的话，将在邮件队列中停留多长时间。QUEUEWARN 选项决定了在通知发件人可能存在投递问题之前，邮件要停留多长时间。前两行把它

www.TopSage.com

们分别设置为 7 天和 4 小时。

接下来的一行设置了隐私标志来禁止 SMTP EXPN（展开地址）命令。参考 18.12.4 节了解有关隐私选项的更多信息。confPRIVACY_FLAGS 选项需要由逗号分隔的值列表。有些版本的 **m4** 要求用两组引号来保护一个有多项的字段中的逗号，但是 Linux 带的 GNU **m4** 会更聪明一些，它不要求有多余的引号：

```
define('confPRIVACY_FLAGS', "noexpn, novrfy")
```

大多数选项的默认值对于一个不太挑剔安全性，或者不太在乎性能的普通站点来说就够了。特别地，默认值会通过关闭中转（relaying）、要求全名地址和要求地址能解析到一个 IP 地址上等方式，试着保护您免受垃圾邮件的侵扰。如果您的邮件枢纽机器非常繁忙，而且要服务许多个邮递列表，那么您可能需要调整一些与性能有关的值。

表 18.11 列出了一些可能需要调整的选项（近 175 个配置选项中的 15%），以及它们的默认值。为了节省空间，显示选项名称时不带它们的 conf 前缀，例如，FALLBACK_MX 选项的名字实际上是 confFALLBACK_MX。我们把该表分成几个子表，以便标识出变量所处理的问题类型：一般、资源、性能、安全性和减少垃圾邮件、以及杂项。有些选项显然适合于多个类别，但是我们只把它们列出一次。

表 18.11 基本的配置选项

	选 项 名	说明以及（默认值）
一般	COPY_ERRORS_TO	出错消息上要 Cc 的地址（无）
	DOUBLE_BOUNCE_ADDRESS	捕获大量的垃圾邮件；有些站点使用**/dev/null**，但是那样做会隐藏严重的问题（postmaster）
资源	MIN_FREE_BLOCKS	接受邮件的最大文件系统空间（100）
	MAX_MESSAGE_SIZE	单个消息的最大字节数（无限）
	TO_*lots_of_stuff*	所有类型事件的超时设定（各有不同）
	TO_IDENT	为检查发件人的身份而进行的身份查询的超时设定；如果为 0，则不进行身份检查（5 秒）
	MAX_DAEMON_CHILDREN	自己进程的最大数目[a]（无限制）
性能	MCI_CACHE_SIZE	在缓存区打开的传出 TCP 连接的数量（2）
	MCI_CACHE_TIMEOUT	保持被缓存的连接打开的时间（5 分钟）
	HOST_STATUS_DIRECTORY	参见 18.13.6 节的介绍（没有默认值）
	FALLBACK_MX	参见 18.13.6 节的介绍（没有默认值）
	FAST_SPLIT	在按收件人分类，并分入多个队列时禁用 MX 查找；参见 18.12.3 节（1 = 真）
	QUEUE_LA	到达这个平均负载值时，就应该把邮件排入队列，而不是例子投递（8×CPU 的数量）
	REFUSE_LA	到达这个平均负载值时就要拒绝邮件（12×CPU 的数量）
	DELAY_LA	到达这个平均负载值时就要减缓投递速度（0 = 没有限制）
	MIN_QUEUE_AGE	作业在队列中必须保留的最短时间；使一台繁忙的机器能更好地处理队列（0）
安全/垃圾邮件	TRUSTED_USERS	用于邮递列表软件的所有者；允许伪造 From 行和重新构造别名数据库（root、daemon、uucp）
	PRIVACY_FLAGS	限制用 SMTP 发出的信息（authwarnings）
	INPUT_MAIL_FILTERS	传入邮件的过滤器列表（空）
	MAX_MIME_HEADER_LENGTH	MIME 信头的最大尺寸（没有限制）[b]
	CONNECTION_RATE_THROTTLE	限制接受邮件连接的速度，减缓 DOS 攻击（没有限制）
	MAX_RCPTS_PER_MESSAGE	减缓垃圾邮件的投递速度；推迟向多出来的收件人发送邮件，而是发一份临时的出错消息（无限）

www.TopSage.com

续表

	选 项 名	说明以及（默认值）
安全/垃圾邮件	DONT_BLAME_SENDMAIL	覆盖 **sendmail** 的安全和文件检查功能；不要随便改动！（safe）
	AUTH_MECHANISMS	Cyrus SASL 的 SMTP 验证机制[c]
其他	LDAP_DEFAULT_SPEC	LDAP 数据库的映射说明，包括服务器正在运行的主机和端口（无定义）

a. 更确切地说是，一次可以运行的子进程的最大数量。当达到上限的时候，**sendmail** 会拒绝连接。这个选项能够防止拒绝服务攻击（DOS）。

b. 这个选项能够防止用户代理缓冲区溢出。“256/128”是一个可用的不错的值——它表示每个信头 256 字节，该信头的每个参数 128 字节。

c. 默认值为“EXTERNAL GSSAPI KERBEROS_V4 DIGEST-MD5 CRAM-MD5”。除非想要降低安全性，否则不要加 PLAIN LOGIN。

18.10 sendmail 中与垃圾邮件相关的功能

垃圾邮件是对广告宣传邮件的专门叫法，通常也称之为不请自来的商业电子邮件。它已经成为一个严重的问题，这主要是因为发件人（至少在美国）不是按字节数付钱，而是纯粹按连接速率付钱的。或者就算他们按字节数付钱，他们也会只发送一则有数千个收件人的消息，然后通过另一台机器来转发它。别的机器支付了按字节算的庞大费用，而发送垃圾邮件者只支付一份拷贝的钱。在许多国家，最终用户按收到的字节数付钱，他们因为不得不为接收垃圾邮件付钱而非常生气。

从市场营销人员的观点看，垃圾邮件的效果不错。响应速度快，费用低廉，并且是即时发送。一份有 3 千万个电子邮件地址的列表只需花费大约 40 美元。

许多发送垃圾邮件的人试图表明他们是无辜的，他们会建议您如果想从他们的邮件列表中消失的话，请用一则写上“请删除”的消息回复他们的电子邮件。虽然他们可能将您删除，但您却刚好替他们验证了一个目前有效的电子邮件地址，这个事实可能会将您放入其他的列表。发送垃圾邮件者还喜欢弄乱他们的邮件信头，试图把谁发送这封邮件，邮件是从哪台机器来的伪装起来。

把电子邮件地址卖给发送垃圾邮件者的家伙近来开始使用字典攻击的形式来搜出未知的地址。扫描软件从一份常用姓氏的清单开始，通过添加不同的大写首字母，期望找出有效的电子邮件地址。为了检查这些地址是否有效，这个软件会连接到，比方说，50 家大型 ISP 的邮件服务器，对无数个地址逐个进行验证（VRFY 或者 RCPT）。

这种探测会对邮件服务器及其向客户投递合法邮件的能力产生巨大影响。**sendmail** 可以通过使用 `PrivacyOption goaway` 对付这种情况，本章将从 18.12.4 节开始介绍这种方法。但是聪明的垃圾邮件软件的精力是非常充沛的，如果 VRFY 被封，它们会尝试 EXPN；如果两者都被封，它们就尝试 RCPT。它们可以用这种方法尝试数百万个地址而从不发送一则消息——但它确实让您的邮件服务器忙得不可开交。

sendmail 有一个选项 `BAD_RCPT_THROTTLE` 能挫败这种行为。如果在邮件信封里被拒绝的地址数量超过了这个选项的值，**sendmail** 就开始在每次被拒绝的 RCPT 命令之后等一秒。

sendmail 已经添加了一些非常好的功能来帮助控制垃圾邮件，这也有助于防范偶尔从邮件衍生的计算机病毒。很遗憾，大多数 ISP 必须传递所有的邮件，所以这些功能相对于服务客户的宗旨来说可能太过苛刻了（当然，或许它们并不算严厉）。不过，在最终用户的站点上使用这些功能可以获得出色的效果。

有 4 种类型的垃圾邮件控制功能。

- 控制第三方或者随意的中继转发（relaying）的规则，中转就是一个站外用户使用您的邮件服务器给另一个站外用户发送邮件。发送垃圾邮件者常常使用转发来试图掩盖他们邮件的真实来源，以避免被他们的 ISP 发现。它还使得他们可以使用您的机时而节省他们自己的。这简

www.TopSage.com

直是杀手。

- 访问数据库，它允许根据地址对邮件进行过滤，这很像一个用于电子邮件的防火墙。
- 黑名单，其中包含了开放的中转站和已知对垃圾邮件友好的站点，**sendmail** 可以对照这些站点进行检查。
- 凭借一种称为 **libmilter** 的通用邮件过滤接口，进行信头检查和输入邮件的过滤。它允许您任意扫描邮件信头和内容，并拒绝与特定模板相匹配的任何消息。

我们将在下面介绍这些新功能，然后考察我们收到的两封垃圾邮件，看看我们应该如何调整邮件系统以便自动地识别和拒收它们。

18.10.1 中继转发

sendmail 和其他邮件传输代理程序要接受传入的邮件，查看信头和信封地址，决定邮件应该去往何处，然后把它传递到适当的目的地。该目的地可以在本地，或者可以是投递链中往下走的另一个传输代理程序。当一则传入的消息没有本地收件人时，就称处理它的传输代理程序担当了一个中继站（relay）的角色。

在 8.9 版的 **sendmail** 之前，默认情况下允许随意转发（也称为开放中继）。**sendmail** 接受在端口 25 上投递给它的任何消息，并尽力完成投递。这样做的前提是 Internet 能睦邻友好。

遗憾的是，发送垃圾邮件者开始滥用中继转发功能，他们利用它来隐瞒自己的身份，更重要的是，他们要使用您的带宽和机时而不是他们自己的。现在的看法是，把邮件服务器配置成开放中继是非常不好的。然而有许多服务器仍然配置成了开放中继。

只有用 RELAY 在访问数据库（参见 18.10.2 节）中做了标记的主机，或者在**/etc/mail/relay-domains** 中列出的主机才允许投递要进行中继转发的邮件。由于对默认行为进行了这样的改动，再加上公众对此认识的提高，使用各种黑名单做前期屏蔽的帮助，开放中继的比例在今后几年里应该降低。

所以，随意的中继转发是很糟糕的。与此同时，有些类型的中继又是很有用的，也是合法的。您怎么分辨应该转发哪些消息，拒收哪些消息呢？中继转发（relaying）实际上只在两种情况下才是必要的。

- 当传输代理程序充当主机的网关，除转发外任何其他方式该主机都不可达时。例如，不一定打开的主机（拨号主机、Windows 的 PC）和虚拟主机。在这种情况下，要中继转发的所有收件人都在同一个域内。
- 当传输代理是其他不那么聪明的主机的传出邮件服务器时。在这种情况下，所有发送主机的主机名或者 IP 地址都在本地（或者至少可以列举出来）。

其他任何似乎要求中继的情形可能都只是因为设计糟糕的缘故（要支持移动用户可能是个例外）。您可以通过指定一台中央服务器接收邮件（客户机访问采用 POP 或者 IMAP）来避免使用中继的第一种情况。第二种情况总是允许的，但是只允许您自己的主机。您可以检查 IP 地址或者主机名，主机易于伪造，但是 **sendmail** 能够核实它们是否是伪造的。

虽然 **sendmail** 在默认情况下把中继转发功能关掉了，但又添加了几项功能可以再把它完全打开，或以某种受限制和约束的方法打开。下面完整地列出了这些功能，但我们建议要当心别开得太大。大多数站点不需要第二张表中列出的任何一项功能，它们确实很危险。下一小节中介绍的 access_db 功能，是允许有限制转发的最安全的方法。

- FEATURE('relay_entire_domain') ——只允许自己的域转发；
- RELAY_DOMAIN('*domain*,…') ——增加其他可转发的域；
- RELAY_DOMAIN_FILE('*filename*') ——与上同，但是从一个文件中获取域列表；
- FEATURE('relay_hosts_only') ——影响 RELAY_DOMAIN，accessdb。

如果使用 SMART_HOST 或者 MAIL_HUB，通过一台特殊的邮件服务器传递邮件，就需要破例了。

www.TopSage.com

该服务器将不得不被设置为转发来自本地主机的邮件。应该这样配置它：

```
FEATURE('relay_entire_domain')
```

同时还做虚拟主机托管的站点可能还需要 RELAY_DOMAIN，才能允许它们的虚拟名字可以进行转发，虽然：

```
FEATURE('use_cw_file')
```

就能为那些域或者主机打开转发功能。

还有其他一些可能性，但是它们充满了问题：

- FEATURE('promiscuous_relay') ——允许所有转发，别用它；
- FEATURE('relay_based_on_MX') ——允许通过 MX 指向您的任何人进行转发；
- FEATURE('loose_relay_check') ——允许“百分号扩展”寻址；
- FEATURE('relay_local_from') ——根据 From 地址进行转发。

promiscuous_relay 功能允许从任何站点转发到任何其他站点。使用它就拿到了落入黑洞名单的单程票。千万不要在 Internet 能公开访问得到的机器上使用这项功能。

relay_based_on_MX 功能不好，因为您不能控制让哪些站点把它们的 MX 记录指向您。典型情况下，只有您自己的主机才会把 MX 记录指向您的邮件服务器，但没有什么能够防止其他站点修改它们的 MX 记录从而指向您。发送垃圾邮件者通常不具有修改 MX 记录的能力，但是阴险的站点肯定可以。

loose_relay_check 功能允许“%扩展”形式（参见 18.2.1 节）的寻址方式，发送垃圾邮件者喜欢使用这种寻址形式。

relay_local_from 功能信任消息信封上的发件人地址，它将转发那些看似来自本地地址的消息。当然，无论是邮件消息的信封还是信头都很容易伪造，而发送垃圾邮件者是伪造专家。

如果您打算以某种形式打开转发，请参阅 **cf/README** 中的 **sendmail** 文档，确保自己不会在不经意间成为发送垃圾邮件者的朋友。当完成之后，请让某个中继检查站点验证您是否在无意中创建了一个开放中继——可以试试 spam.abuse.net。

有一些发生匹配错误的配置会使您的主机可能转发怪异的地址，这些地址是 UUCP 寻址句法的误用。当然，如果您没有 UUCP 连接，可以使用：

```
FEATURE('nouucp', 'reject')
```

来预先防止这种可能性。当前的 **sendmail** 默认不支持任何古老的连网技术，比如 UUCP、BITNET 或者 DECnet。

另一种常见的转发是 LUSER_RELAY，它用于并不存在的本地用户。它默认按照下面这样定义：

```
define('LUSER_RELAY', 'error:No such user')
```

sendmail 配置错误的站点有时会在 Internet 上向外泄漏不全的本地用户名（通常是在 Cc 行上）。试图给这类邮件写回复的人会把回复寄给一个显然不存在的本地用户。这种转发通常被称为“失败者转发”，并被指向了错误的邮寄程序。

除非您想要返回一则不同的消息或者实现某种特殊的处理，否则不需要改变这个配置。有些站点把“失败者”的邮件发给模糊匹配的一个人或者一个程序，以防发件人打错了，或者只是登录名稍有错误。

18.10.2 访问数据库

sendmail 中包括了对访问数据库的支持，您可以用它来为自己的站点建立一个专门用于邮件的防火墙。它检查来自外部世界的邮件，如果邮件来自特定用户或者域就拒收。还可以使用访问数据库来

www.TopSage.com

TopSage.com

指定一台机器愿意为哪些域转发邮件。

用下面一行启用访问数据库（access database）：

```
FEATURE('access_db', 'type filename')
```

如果没有指定 *type* 和 *filename*，数据库默认为从文件**/etc/mail/access** 建立的 hash 类型（如果使用 DB 数据库——这取决于 DATABASE_MAP_TYPE 的设置）。DBM 数据库不使用 *type* 字段。和往常一样，用 **makemap** 创建数据库：

```
# makemap hash /etc/mail/access < /etc/mail/access
```

access 文件的关键字段可以包含电子邮件地址、用户名、域名或网络编号。

例如：

```
cyberspammer.com          550 Spam not accepted
okguy@cyberspammer.com    OK
badguy@aol.com            REJECT
sendmail.org              RELAY
128.32                    RELAY
170.201.180.16            REJECT
hotlivesex@               550 Spam not accepted
friend@                   550 You are not my friend!
```

值的部分必须包含表 18.12 所示的项之一。取 RELAY 这个值是最常用的，它只是接受消息，并把它转发给它的最终目的地。OK 接受消息，但是不允许做中继转发。REJECT 根本不接受消息。SKIP 能让您允许有例外。例如，如果想要中转一个域内除了两台主机之外所有主机的邮件，就可以用 SKIP 动作列出这台主机，然后用 RELAY 动作列出这个域。前后顺序没有关系。

表 18.12 访问数据库的值字段中能出现的内容

值	作 用
OK	接受邮件并正常投递它
RELAY	按照地址接受邮件并把它中继转发到它的目的地，启动逐个主机的中继转发功能
SKIP	允许普遍规则有例外
REJECT	用一则普通的出错消息拒收邮件
DISCARD	悄悄地丢弃这条消息
FRIEND	用于垃圾邮件，由 delay-checks 功能使用。如果匹配，跳过其他的信头检查
HATER	用于垃圾邮件，由 delay-checks 功能使用。如果匹配，应用其他的信头检查
xxx message	返回一条出错信息，*xxx* 必须是 RFC 821 中规定的数字代码[a]
ERROR:*xxx message*	同上，但清楚地标出是一则出错消息
ERROR:*x.x.x message*	*x.x.x* 必须是一个遵守 RFC1893 的投递状态通知（550 出错代码的推广）

a. 例如，550 是单个错误的代码。

这个数据库文件将允许接收来自 okguy@cyberspammer.com 的消息，但是拒收其他所有来自 cybersammer.com 的邮件，并返回预设的错误消息。从 sendmail.org 或 128.32.0.0/16（加州大学伯克利分校的网络）来的邮件将被转发。从 badguy@AOL 和从任何域的 hotlivesex 或 friend 来的邮件将被拒收。

采用冒号分隔形式的 IPv6 地址也可以用在项的左边，但是必须用“IPv6:”打头。用户名 hotlivesex 和 friend 之后的@是必要的，用来把它们同域名加以区别。

550 是一个 RFC821 错误代码。RFC1893 错误代码（或者把它们称为“投递状态通知消息”）更为广泛一些。第一个数字是 4 则表示是个临时性错误；5 则说明它是一个永久性错误。在表 18.13 中列

www.TopSage.com

出了一些错误代码。

表 18.13 RFC1893 投递状态代码

临时错误	永久错误	含义
4.2.1	5.2.1	邮箱被禁用
4.2.2	5.2.2	邮箱已满
4.2.3	5.2.3	消息太长
4.2.4	5.2.4	列表扩展问题
4.3.1	5.3.1	邮件系统已满
4.4.4	5.4.4	不能路由
4.4.5	5.4.5	邮件拥塞
4.7.*	5.7.*	违反站点政策

为了进行更好的控制，关键字字段（左边）可以包含标记 Connect，To 和 From，以及在 8.12 以后的版本里的 Spam，来控制应用过滤器的方式。Connect 是指连接信息，比如客户机 IP 地址或客户机的名字(host-name)。To 和 From 是指信封地址而不是信头地址。Spam 标记通过“垃圾邮件朋友(spam firend)”和“垃圾邮件憎恨者（spam hater)”测试对全局规则有例外处理。用 delay_checks 可以启用这一功能。

```
FEATURE('delay_checks',  'friend')
FEATURE('delay_checks',  'hater')
```

如果 access_db 中有一项和映射关系右边的 FRIEND 匹配，那么第一个功能就跳过其他可能拒收消息的规则集。如果 access_db 的值是 HATER，那么第二个功能就施加其他的规则集。这 4 个标记能够在更细的粒度上控制邮件的中继转发和拒收，它们也可以用来覆盖其他的限制。可以用针对垃圾邮件的 FRIEND 或者 HATER 标记来满足对您在全站范围内的垃圾邮件策略有抱怨的用户。

如果使用了某个标记，那么将先尝试带标记信息查找，然后不带标记信息进行查找，以保持对较老的访问数据库的向后兼容性。

下面是一些例子：

```
From:spammer@some.domain    REJECT
To:good.domain              RELAY
Connect:good.domain         OK
Spam:abuse@                 FRIEND
```

从 spammer@some.domain 来的邮件将被封锁，但您仍然可以给这个地址发邮件，即使它在黑名单上。邮件将被转发到 good.domain，而不能转发从它那里来的邮件（假设已在别处禁止转发)。从 good.domain 来的连接是允许的，即使它被列在一个基于 DNS 的拒收名单中。向 abuse@localdomain 发送邮件可以通过，哪怕是从 spammer@some.domain 来的发往 abuse@localdomain 的邮件也可以通过，访问数据库的第一行已经拒收了 spammer@some.domain 的邮件。

许多站点使用访问数据库来控制垃圾邮件或者做管制。我们在科罗拉多大学计算机科学系的传入邮件主机通过使用 access_db 功能可以拒收从超过 500 个已知的发送垃圾邮件者发来的邮件，这些发送垃圾邮件者是根据地址、域或者 IP 网络识别出来的。

18.10.3 把用户或站点列入黑名单

如果希望封锁发往本地用户或主机的邮件，请使用:

www.TopSage.com

```
FEATURE('blacklist_recipients')
```

它支持访问文件中以下类型的项：

```
To:nobody@                  550 Mailbox disabled for this user
To:printer.mydomain.edu     550 This host does not accept mail
To:user@host.mydomain.edu   550 Mailbox disabled for this user
```

这几行将封锁发给任何主机上的 nobody 用户、发往主机打印机和某台机器上特定用户地址的传入邮件。使用 To:标记能让这些用户发送消息，只是不能接收邮件，有些打印机就有这个能力。

遗憾的是，现在实际上不可能手工维护一个像这样的黑名单。还好，通过 DNS 可以得到由几个团体维护的黑洞列表，其中有些可以免费提供。

要把 DNS 风格的黑洞名单包括进来，可以使用 dnsbl 功能：

```
FEATURE('dnsbl', 'sbl-xbl.spamhaus.org')
```

这项功能使得 **sendmail** 拒收来自任何站点的邮件，只要其 IP 地址被列在“Spamhaus Block List”中，该名单中列出的是已知的垃圾邮件源，在 sbl-xbl.spamhaus.org 维护。其他列表给已知的通过拨号发送垃圾邮件者和运行开放中继的站点分了类，这些地方都可能成为发垃圾邮件者的天堂。

这些黑名单是通过 DNS 系统的一种巧妙用法进行发布的，因此起名叫 dnsbl。例如，把一个特殊形式的 DNS 资源记录：

```
IP-address.sbl-xbl.spamhaus.org    IN    A    127.0.0.2
```

放入 blackholes.mail-abuse.org 域的 DNS 数据库，那么在启用 dnsbl 功能时，来自该主机的邮件将被阻塞（因为 sendmail 将显示地进行检查，以弄清是否存在这样一条记录）。本例中的 *IP-address* 是一个主机地址，它用 4 个由点号连接的小于 255 的数字代表 IP 地址，其中八位组的顺序是颠倒的。

您可以数次包括 dnsbl 功能，以检查不同列表中的滥发邮件者：只需添加第二个参数以指定黑名单名字服务器，第 3 个参数说明您希望返回的错误消息。如果省略第 3 个参数，返回的固定错误消息将来自包含该记录的 DNS 数据库。

18.10.4 信头检查

发垃圾邮件的人经常会企图隐藏他们的身份。从 **sendmail** 8.9 起，如果信封的 From 地址不是 *user@valid.domain* 的形式，那么就会拒收邮件。可以用下面的功能来放弃这种做法：

```
FEATURE('accept_unresolvable_domains')
FEATURE('accept_unqualified_senders')
```

这两种功能中的第一种功能可以接受从根本不存在的域或者不能在 DNS 名字树中解析的域来的邮件。第二个功能允许 From 地址只有一个用户名，不带主机或者域名。除非在防火墙的后面，而且那里只能有本地 DNS 数据，否则就不要使用这两种功能中的任何一种。如果您发现自己想要把这两种功能打开，或许应该代之以考虑一下重新设计 **sendmail** 和 DNS 环境。要求有合法的信封发件人地址，就能大大减少垃圾邮件。

对信头做详细检查是一种强有力的反垃圾邮件机制，它能利用底层的 **sendmail** 配置文件语法，我们在这里不会介绍这些语法。通过使用信头检查机制，**sendmail** 就能查看信头内是否有指定的模式存在（例如，“To: friend@public.com”），并且在邮件被投递到用户的邮箱之前拒收它们。

信头检查还可以用于识别电子邮件携带的病毒，它们具有某种有特色的信头行。例如，1999 年的

www.TopSage.com

Melissa 病毒包含的主题行是“Important Message From…”。在 Melissa 病毒发布和被识别后的几个小时内，sendmail.com 就贴出了一个本地规则集用来辨别和丢弃这种病毒。

当病毒的特征是有特点的、容易用 **sendmail** 规则表示的时候，sendmail.com 通常会很快（在 sendmail.com 和 sendmail.org 网站上同时）贴出它的补丁。

要得到针对垃圾邮件和病毒所用的过滤器规则的代表样本，请参见 Eric Allman 的个人主机 knecht 上用的 **sendmail** 配置。这项配置作为**/cf/cf/knecht.mc** 包括在 **sendmail** 的软件发布中。把这个垃圾邮件过滤器规则拿过来，并把它们添加到您的**.mc** 文件的末尾。

在查看各个例子时，我们已经看到信头检查规则可用于：

- 发给域 public.com 中任何用户的邮件；
- 发给“friend”或“you”的邮件；
- 邮件主题为“Important Message From…”（Melissa 病毒）；
- 邮件主题为“all.net and Fred Cohen…”（Papa 病毒）；
- 邮件主题为“ILOVEYOU”（iloveyou 病毒及其变体）；
- 数以亿计的市场推销垃圾邮件；
- 带有被破坏的 Outlook Express 信头的邮件（SirCam 蠕虫）。

所有的信头检查规则都位于**.mc** 配置文件末尾的 `LOCAL_CONFIG` 和 `LOCAL_RULESETS` 语句之下。借助于 **m4** 的 divert 命令的帮助，**sendmail** 就知道了把它们放到原始配置文件中的什么位置。

您所实现的减少任何垃圾邮件的措施或多或少都会挡住一些发送垃圾邮件者，同时也提高了其他发送垃圾邮件者的门槛水平。请使用 `error` 邮寄程序发出“user unknown（未知用户）”错误消息而不要使用 `discard` 邮寄程序，因为许多发送垃圾邮件者会清理他们的列表。干净的列表会更有价值，所以如果可以拦截垃圾邮件，过滤它并用一则错误消息回复，也许就能从一些列表中被删除。

18.10.5 速率和连接限制

sendmail 8.13 增加了一个 `ratecontrol` 功能，它可以一个主机一个主机地，或者一个网络一个网络地设定限制，限制接受从这些地方来的传入连接的速率。对于无法在不造成问题的前提下彻底封锁垃圾邮件源的地方（像拨号用户比例很高的大型 ISP），这项限制特别有助于放慢垃圾邮件的速度。要启用速率控制，可以在**.mc** 文件里加上如下一行：

```
FEATURE('ratecontrol', 'nodelay', 'terminate')
```

此外，您还必须在**/etc/mail/access** 文件里列出要被控制的主机或者网络及其限制阈值。例如，下面两行：

```
ClientRate:192.168.6.17 2
ClientRate:170.65.3.4 10
```

就限制 192.168.6.17 和 170.65.3.4 这两台主机每分钟分别只能有 2 次和 10 次新连接。

`conncontrol` 功能对并发连接数做了类似的限制。您可以在**.mc** 文件里用下面这样一行配置启用该功能：

```
FEATURE('conncontrol', 'no delay', 'terminate')
```

和 `ratecontrol` 一样，您可以在 `/etc/mail/access` 文件里指定要限制的主机和网络：

```
ClientConn:192.168.2.8 2
ClientConn:175.14.4.1  7
```

www.TopSage.com

```
ClientConn:             10
```

上述配置的结果为：限制 192.168.2.8 的并发连接数为 2，175.144.4.1 的并发连接数为 7，而所有其他主机的并发连接数为 10。

18.10.6 Slamming

在 8.13 里的另一项很好的功能是 greet_pause。当一个远程的 MTA 连接到您的 **sendmail** 服务器时，SMTP 协议要求它要等收到您服务器的欢迎消息之后才开始通信。不过，发垃圾邮件的程序（和蠕虫/病毒）往往立即就会发一条 EHLO/HELO 命令。这种现象可以部分解释为发垃圾邮件的工具里的 SMTP 实现得很差，但也可能是一项替发垃圾邮件的人节省时间的特性。无论其真实原因是什么，这种做法都值得怀疑，并称为“slamming”。

greet_pause 功能让 **sendmail** 在开始连接的时候先等上指定的一段时间，再向它新发现的朋友发问候消息。如果远程的 MTA 没有等到得到正常的问候，就在这段有意放慢响应的时间内继续发一条 EHLO 或者 HELO 命令，那么 **sendmail** 就在日志里记下一则出错记录，并且拒绝该远程 MTA 后续的命令。

您可以在**.mc** 文件里加上如下一行启用 greet_pause 功能：

```
FEATURE('greet_pause', '700')
```

这行配置让每次新连接开始的时候都延迟 700 毫秒。也可以像 connctrol 和 ratecontrol 那样，逐个主机或者逐个网络来设置，不过大多数站点上的这个功能都使用统一的延迟值。

18.10.7 邮件过滤

8.12 版的 **sendmail** 引入了一种对信头过滤机制的推广，能形成一种非常有效的反垃圾邮件工具。它是一个邮件过滤 API（application programming interface，应用编程接口），人们可以使用它来开发他们自己的邮件过滤程序。这些过滤程序位于 **sendmail** 和传入消息之间，能够用来识别出病毒或者垃圾邮件的特征，丢弃它或者对它做日志记录（或者采取您觉得合适的任何操作）。过滤的目标既可以是元数据也可以是消息内容。

邮件过滤（mitering）既有可能成为一种用来反垃圾邮件的强有力工具，也有可能成为侵犯用户隐私的工具。想要准确掌握公司通过电子邮件流传出去了哪些信息的经理们可能会成为最早采用这种机制的人。8.12 版的 **sendmail** 不能将邮件过滤功能用在传出的邮件上，但是它已经列入了“to do”工作表。

邮件过滤库叫做 **libmilter.sendmail**，用 INPUT_MAIL_FILTER 或者 MAIL_FILTER 配置指令来调用输入过滤机制，而用名为 MILTER_MACROS_*的选项控制邮件过滤的动作，对 SMTP 会话的每个阶段所应用的过滤器做细粒度的控制。

例如，下面这一行：

```
INPUT_MAIL_FILTER('filtername', 'S=mailer:/var/run/filtername.socket')
```

会把每个传入的消息通过第二个参数所指定的套接口传给程序**/etc/mail/filtername**。

参见 **sendmail** 软件发布里的 **libmilter/README** 或者 **libmilter/docs** 目录下的 HTML 文档可以了解到更多的信息。README 文件对 miltering 进行了概述，并且给出了一个过滤器的简单例子，它把消息的日志记入一个文件。**docs** 下面的文件介绍了库的接口，并且说明了如何使用各种调用来构建您自己的邮件过滤程序。

www.TopSage.com

大家网 TopSage.com

18.10.8 处理垃圾邮件

反垃圾邮件也许是一种充满困难和挫折的工作。以往，这样的工作成效不大。即使有成堆的垃圾邮件穿过了您的反垃圾邮件屏障，也不要受找出个别发送垃圾邮件者的诱惑。花时间分析垃圾邮件信头和为发送垃圾邮件者头疼纯粹是浪费时间。没错，这样做是打胜仗的态度，但是在这些问题上花时间也许并不能减少进入站点的垃圾邮件的数量。

向固定发送垃圾邮件者的 ISP 举报可以很快抓住他们，但打一枪换一个地方、每个 ISP 账号只使用一次就放弃的发送垃圾邮件者就很难统计了。如果他们为某个网站做广告，那么该网站是有责任的；如果它是一个电话号码或邮政地址，就更难找出罪犯了，但也不是不可能。许多流动的发送垃圾邮件者似乎基本上都逃脱了处罚。

各种黑洞列表已经对阻断垃圾邮件起到了一些作用，并极大地减少了开放中继的数目。被列入黑名单会对业务产生严重影响，所以一些 ISP 和公司会谨慎地控制他们的用户。

我们关于垃圾邮件的主要建议是，请使用预防性措施和可用的公开维护的黑名单。还有另一种可能，把寄到您网点的电子邮件都转寄到一家做外包的反垃圾邮件公司，例如 Postini。不过，这么做可能会在性能、隐私或者可靠性上面有一些损失。

告诫您的用户只需删除他们收到的垃圾邮件即可。许多垃圾邮件的消息中包含有收件人如何才能从邮件列表中将自己删除的指示。如果您遵循那些，发送垃圾邮件者可能把您从当前列表中删除，但他们会立即将您添加到其他的若干个列表，并注明“能到达一个会读到这个消息的真人”。然后您的电子邮件地址甚至会更值钱。

如果您喜欢加入到反垃圾邮件的阵营中，一些网站可以帮助您。一个好网站是 www.abuse.net。另外两个知名的 Web 站点是 spamcop.net 和 cauce.org。SpamCop 有一些工具有助于分析邮件信头，并判断真实的发件人是谁。cauce.org 这个站点上有垃圾邮件相关法规的信息。在美国，您可以让联邦贸易委员会（Federal Trade Commission）提供帮助。

18.10.9 SpamAssassin

SpamAssassin 是一个过滤器（可以通过 **sendmail** 的 milter 调用它），能高效地识别垃圾邮件。它使用一种积点系统来评估一则消息的错误之处。它能捕获所有真正的垃圾邮件，很少误报。如果一条消息累积的积点太多了（可以全站配置，也可以逐个用户单独配置），SpamAssassin 就给这则消息打个标记。接下来，通过运行一个服务器端过滤器（例如 Cyrus 的 **sieve**），或者配置您自己的用户代理，您就可以把可疑的消息送到一个垃圾邮件文件夹里。您甚至能教会 SpamAssassin 什么是好的消息，什么是坏的消息。在您设置 SpamAssassin 和调整它的参数时，要仔细检查所有的垃圾邮件。从 spamassassin.apache.org 可以得到这个软件。

18.10.10 SPF 和 Sender ID

反垃圾邮件最好的方法就是从源头堵住它。这听上去容易做起来难。Internet 的结构使得难以跟踪消息的来源和验证其身份。人们需要一种着实有效的途径来验证发送邮件的实体名副其实。

许多提案已经解决了这个问题，但是 SPF 和 Sender ID 最为行之有效。SPF（或者说 Sender Policy Framework，发件人政策框架）现在由 IETF 确定为标准（RFC 4408）。它定义了一组 DNS TXT 记录（参考 15.7.11 节的内容），一家单位可以借助这些记录标识出其“正式的”对外邮件中继。于是，如果电子邮件不是从这些正式来源发出的，那么 MTA 就可以拒绝从这家单位的域内发来的任何邮件。当

www.TopSage.com

然，只有当大多数单位都公布了 SPF 记录，这种办法才能取得好的效果。能够下载到几种邮件过滤库（milter），它们在 **sendmail** 中实现了这个功能。

Sender ID 和 SPF 从形式和功能上看，实质上是等价的。不过微软掌握了 Sender ID 关键部分的专利，所以它已经成了一个颇具争议的话题。在撰写本书（英文版）的时候，微软仍然在尝试迫使业界采用它的专有标准。

18.11 配置文件举例

作为学习在现实世界里如何配置 **sendmail** 的一个例子，我们来看一个很小但是非常了解 **sendmail** 的公司——Sendmail，Inc.（Sendmail 有限公司）——所用的配置文件。他们的邮件系统设计方案包括一台主邮件枢纽主机（master mail hub），它同时负责传入邮件和发出邮件的处理任务。传入的所有邮件都由它接受，并立即传送给一组内部的 IMAP 服务器，这些服务器在把每则消息投递到用户的邮箱之前，先要检查病毒。邮件枢纽主机也能检查每一条传出的消息是否有病毒，这样一来，Sendmail，Inc.就不会因为通过电子邮件散布病毒而负责任。我们先看客户的配置，然后再查看更复杂的邮件主机。

在这些例子中，我们都对原始文件进行了轻微修改，删去版权说明，添加临时注释，并删除行尾的 **m4** `dnl` 指令。如果您把我们的任何例子用作自己的**.mc** 文件的模板，请一定要删除行尾的注释。

18.11.1 sendmail.com 的邮件客户机

用于客户机的 **smi-client.mc** 文件相当简单。它使用邮件主机 smtp.sendmail.com，这其实只是机器 foon.sendmail.com 的一个别名。使用一条 MX 记录（或者一条 CNAME[31]记录）来指向邮件服务器是个好主意，当您希望移动自己的邮件主机时，这样改起来很容易。

请注意这个文件中的日期是 1998 年 10 月。自那时以来 **sendmail** 已经升级了许多次，但这个配置文件却不需要修改。

```
divert(-1)
#####  This file contains definitions for a Sendmail,
#####  Inc. client machine's .mc file.
divert(0)
VERSIONID(`@(#)smi-client.mc 1.0 (Sendmail) 10/14/98')
OSTYPE(`bsd4.4')
FEATURE(`nocanonify')
undefine(`ALIAS_FILE')
define(`MAIL_HUB', `smtp.sendmail.com')
define(`SMART_HOST', `smtp.sendmail.com')
define(`confFORWARD_PATH', `')
MAILER(`local')
MAILER(`smtp')
```

`MAIL_HUB` 和 `SMART_HOST` 行把传入和发出的邮件定向到主机 smtp.sendmail.com。对 DNS 中的 MX 记录应该予以配合，列出比单个客户机拥有更高优先级的主机（在 MX 记录中的编号较低）。**.forward** 文件的路径被设置为空，别名文件也被设置为空，所有别名扩展都在这台主邮件枢纽主机（master）上进行。在这里指定了 `nocanonify` 功能以节省时间，因为无论如何在主邮件枢纽主机（master）上都要进行 DNS 查找的。

[31] MX 记录实际上比 CNAME 记录更有效，CNAME 要求对实际的名字做第二次查询才能得到 IP 地址。

www.TopSage.com

18.11.2 sendmail.com 的邮件主机

sendmail.com 的主邮件枢纽主机（master）可能是被攻击得最多的 **sendmail** 软件安装之一。它必须在攻击者们搞出的所有狡猾的邮寄程序的攻击之下保证安全，并保护它身后的机器。下面是它的配置文件：

```
divert(-1)
# Created with Sendmail Switch, sendmail.com's commercial product.
divert(0)
ifdef(`COMMERCIAL_CONFIG', `INPUT_MAIL_FILTER(`mime-filter', `S=local:
    /var/run/mime-filter/mime-filter.sock')')
LDAPROUTE_DOMAIN(`sendmail.com sendmail.net sendmail.org')
MASQUERADE_AS(`sendmail.com')
MASQUERADE_DOMAIN(`sendmail.com')
RELAY_DOMAIN(`sendmail.com sendmail.net sendmail.org')
define(`MAIL_HUB', `internal-hub.sendmail.com')
define(`QUEUE_DIR', `/var/spool/mqueue/q*')
define(`SMART_HOST', `virus-scan.sendmail.com')
ifdef(`COMMERCIAL_CONFIG', `define(`confCACERT', `/local/certs/cacert.pem')')
ifdef(`COMMERCIAL_CONFIG', `define(`confCACERT_PATH', `/local/certs/trustedcerts')')
define(`confCHECK_ALIASES', `True')
ifdef(`COMMERCIAL_CONFIG', `define(`confCLIENT_CERT', `/local/certs/cert.pem')')
ifdef(`COMMERCIAL_CONFIG', `define(`confCLIENT_KEY', `/local/certs/key.pem')')
define(`confEIGHT_BIT_HANDLING', `mimify')
define(`confLDAP_DEFAULT_SPEC', ` -h "ldap.sendmail.com ldap2.sendmail.com"
    -b "dc=sendmail,dc=com" -p 1389')
define(`confREFUSE_LA', `99')
define(`confRUN_AS_USER', `mailnull')
ifdef(`COMMERCIAL_CONFIG', `define(`confSERVER_CERT', `/local/certs/cert.pem')')
ifdef(`COMMERCIAL_CONFIG', `define(`confSERVER_KEY', `/local/certs/key.pem')')
define(`confTO_IDENT', `0s')
define(`confTO_QUEUEWARN', `2d')
ifdef(`confPOP_TO', `', `define(`confPOP_TO', `900')')
FEATURE(`accept_unqualified_senders')
FEATURE(`accept_unresolvable_domains')
FEATURE(`allmasquerade')
FEATURE(`always_add_domain')
FEATURE(`domaintable')
FEATURE(`ldap_routing', `ldap -1 -v mailHost -k ldap -1 -v mailhost -k
    (&(objectclass=mailRecipient)(|(mail=%0)(|(mailAlternateAddress=%0))))',
    `ldap -1 -v mail -k
    (&(objectclass=mailRecipient)(|(mailalternateaddress=%0)))', `passthru')
FEATURE(`mailertable')
FEATURE(`masquerade_entire_domain')
FEATURE(`masquerade_envelope')
FEATURE(`relay_entire_domain')
FEATURE(`use_cw_file')
MAILER(`local')
MAILER(`smtp')

LOCAL_RULESETS
SLocal_check_rcpt
R$*    $: $&{verify}
ROK    $# OK
```

主邮件枢纽主机（master）的任务就是把传入的邮件路由到正确的内部服务器上，并且作为传出邮件的中继转发主机。因为有下面两行：

```
FEATURE('accept_unqualified_senders')
FEATURE('accept_unresolvable_domains')
```

故主邮件枢纽主机（master）要接受传入的所有邮件，哪怕邮件是从没有全名的发件人和不能解析的域发来的也要接受。这样一来，把 **sendmail** 或者 DNS 配错了的客户仍然有可能通过。这些规则撤销了能够捕捉大量伪造信头的垃圾邮件的默认设置。身份识别功能被关闭了（超时时间设为 0），从而加快了投递传入邮件的速度。

这台主邮件服务器首先检查传入的消息是否有某些类型的 MIME 附件，这些附件经常被病毒利用（INPUT_MAIL_FILTER 语句）。那里调用的 **mime-filter** 包含下面这样的几行：

```
*:anniv.doc:* error:Your email was not accepted by Sendmail, it appears to be
    infected with the Melissa-X virus.
*:.vbs:*        error:For security and virus protection reasons, Sendmail does
    not accept messages with VBS files attached.  Please retransmit your
    message without the VBS file.
```

MIME 附件的**.vba**、**.dot**、**.exe**、**.com**、**.reg** 等类型都被拒绝，但是这里并没有做全面的病毒扫描，因为这样做会减慢传入邮件的处理速度。主邮件枢纽主机（master）使用 LDAP（带有一个针对站点的特定方案）来查找每个消息的收件人，并把它传送到正确的内部 IMAP/POP 服务器上。如果没有在 LDAP 数据库中找到收件人，那么邮件就被发送给一台内部的主机（MAIL_HUB 语句）做进一步的处理。IMAP/POP 服务器和内部主机都会在把消息投递到用户的邮箱之前对它做全面的病毒扫描处理。

根据客户机上的 SMART_HOST 语句，传出的邮件也通过这台主邮件枢纽主机（master）进行发送。要通过邮件服务器 sendmail.com 发送一则消息，在 sendmail.com 域之外的机器必须提供一个由 sendmail.com 证书服务签发的证书。本公司雇员在到客户方的现场工作时，就可以采用这种机制通过 sendmail.com 向第三方中转（relay）发送邮件，而其他人却没有办法这样做。这种约定验证每个用户的身份，防止假造的邮件经 sendmail.com 中转发出。

在接受了发往 Internet 的邮件之后，这台主机（master）就把它传给 SMART_HOST 进行病毒检查。主邮件机器不太忙，可以自己来完成病毒扫描工作，但是如果在那里完成扫描的话，在邮件确实发出之前，发出邮件的用户不得不等着扫描完成。把它排入队列进行病毒扫描的机器会让用户更惬意一些——他们觉得消息似乎是转眼就发出去了。

在配置文件结尾处的 LOCAL_CONFIG 的几条规则就是通常检查信头、查找各种病毒和已知的发送垃圾邮件者的地方。在 **sendmail** 软件发布中的样本文件 **knecht.mc** 里，就能找到很好的例子。我们在下面包含了一个样本。

在 2001 年夏天，颇具破坏性的 SirCam 蠕虫到处肆虐。下面的片段是从 **sendmail** 软件发布中的 **knecht.mc** 文件里节录出来的，它能捕获这个蠕虫病毒。SirCam 是首次能够产生随机信头内容的恶意病毒之一。如果不是它的作者犯了一个错误，使 SirCam 消息和真正的 Outlook Express 消息有了区别，那么用通常的工具都抓不到它。消息的内容相当常见（它请您就附带的附件给出意见），因此成了 8.12 版 sendmail 中新加的 **libmilter** 过滤机制的使用对象。在软件世界里没有产品级的可靠保障，似乎解决所有这些 Microsoft 病毒和蠕虫的做法就是卸载 Windows 并安装 Linux！参考 18.10.7 节了解有关 **libmilter** 的更多信息。

```
LOCAL_RULESETS

KSirCamWormMarker regex -f -aSUSPECT multipart/mixed;boundary=----
    .+_Outlook_Express_message_boundary
HContent-Type: $>CheckContentType

SCheckContentType
R$+             $: $(SirCamWormMarker $1 $)
RSUSPECT        $#error $: "553 Possible virus, see http:
    //www.symantec.com/avcenter/venc/data/w32.sircam.worm@mm.html"
```

www.TopSage.com

```
HContent-Disposition:$>CheckContentDisposition

SCheckContentDisposition
R$-             $@ OK
R$- ; $+        $@ OK
R$*             $#error $: "553 Illegal Content-Disposition"
```

在 sendmail.com 公司客户机上，它们自己的配置文件里都没有垃圾邮件控制机制，因为所有传入这个站点的邮件都是通过外部的邮件枢纽主机和一台内部的枢纽主机来传送的，垃圾邮件都在那里给挑出来了。本例中的有些功能和其他结构没有在我们介绍配置的章节里讲过，但是可以在 **cf/README** 文件里找到它们的文档说明。

18.12 安全与 sendmail

随着 Internet 的爆炸性增长，像 **sendmail** 这样接受用户所提供的任意输入并把它投递给本地用户、文件或 shell 的应用常常会为黑客们提供一条实施攻击的坦途。**sendmail** 和 DNS 甚至 IP，正在尝试把身份验证和加密作为内建的解决方案以解决一些基本的安全性问题。

最近美国关于加密技术的出口法规开始松动，这使得 **sendmail** 在发布时可以内建对加密功能的挂接接口。版本 8.11 及其后继版本都支持用 SSL（Secure Socket Layer，安全套接口协议层）进行 SMTP 身份验证和加密，SSL 也称为 TLS（Transport Layer Security，传输层加密）。**sendmail** 在这里使用术语 TLS，并且已经把它作为对 SMTP 协议的一种扩展——STARTTLS——实现了。TLS 给它带来了 6 个用于鉴定文件和给文件上锁的新配置选项。对访问数据库进行匹配的新操作要求身份验证必须先取得成功。

在本节中，我们将描述 **sendmail** 的权限模型、所有权和隐私保护的演变。然后简要讨论 SASL（Simple Authentication and Security Layer，简单身份验证和加密层协议）和它在 **sendmail** 中的使用。

随着时间的推移，**sendmail** 已经逐渐加强了它的安全性，现在，在它相信一个文件（比如说 **.forward** 或 **aliases**）之前，它对文件权限是非常挑剔的。虽然这种安全性的加强通常会受到欢迎，但有时也有必要放松强硬的新规则。为了达到这个目的，**sendmail** 引入了 `DontBlameSendmail` 选项，这样命名是希望这个名字会提醒系统管理员，他们正在做被大家认为是不安全的事情。

这一选项有许多可能的值——最新统计有 55 种之多。默认值是 `safe`。值的完整列表请参见 **sendmail** 软件发布里的 **doc/op/op.ps** 这个文件。还是继续把这个选项设置为 `safe` 吧。

18.12.1 所有权

在 **sendmail** 世界中有 3 个用户账号很重要：`DefaultUser`、`TrustedUser` 和 `RunAsUser`。

默认情况下，除非邮寄程序的标志位另行指定，否则所有 **sendmail** 的邮寄程序都作为 `DefaultUser` 运行。如果在 **/etc/passwd** 文件中存在用户 mailnull、sendmail 或者 daemon，那它就是 `DefaultUser` 了。否则，它默认为 UID 1 和 GID 1。我们推荐使用 mailnull 账号和一个 mailnull 组。把它添加到 **/etc/passwd** 中，用一个星号作为口令，没有有效的 shell，没有主目录，默认属组为“nogroup”。您也要在 **/etc/group** 文件里加上 mailnull。Mailnull 账号不应该拥有任何文件。如果 **sendmail** 不是以 root 身份运行的，那么邮寄程序必须设置 setuid 位。

如果设置了 `RunAsUser`，那么 **sendmail** 会忽略 `DefaultUser` 的值，而以 `RunAsUser` 的身份完成所有工作。如果您正在以 setgid（到 smmsp）方式运行 **sendmail**，那么提交邮件的 **sendmail** 只是通过 SMTP 把消息传递给真正的 **sendmail**。真正的 **sendmail** 没有设置它的 setuid 位，但它的启动脚本需以 root 身份运行。

www.TopSage.com

sendmail 的 TrustedUser 可以拥有映射和别名文件。TrustedUser 可以启动守护进程或重建 **aliases** 文件。之所以存在这个功能多是为了支持 **sendmail** 的 GUI 界面，需要这样的界面来向有些用户提供受限的管理控制权。如果您设置了 TrustedUser，那么就要保护好它所指向的账号，因为这个账号很容易被利用，得到 root 权限。TrustedUser 不同于称为 TRUSTED_USERS 的 **sendmail** 类，后者决定谁可以重写消息的 From 行[32]。

sendmail 打开到端口 25 的套接口连接之后，以 RunAsUser 作为 UID 来运行。编号小于 1024 的端口只能由超级用户打开，因此，**sendmail** 最初必须作为 root 运行。不过，执行完这项操作之后，**sendmail** 可以切换到另一个 UID。如果 **sendmail** 被欺骗做一些不好的事，这种切换就降低了损害或访问的风险。不要在支持用户账号或者其他服务的机器上使用 RunAsUser 功能，它是只供防火墙或者堡垒主机使用的。

在默认情况下，**sendmail** 不切换身份并继续作为 root 运行。如果您把 RunAsUser 改为不是 root 的其他身份，那么还必须修改其他几个地方。RunAsUser 必须拥有邮件队列，能够读取所有的映射和包含文件，能够运行程序等。估计要花几个小时才能找出所有必须修改的文件和目录的所有权关系。

18.12.2 权限

文件和目录权限对 **sendmail** 的安全性十分重要。请使用表 18.14 中列出的设置以确保安全。

表 18.14 sendmail 相关目录的属主和权限

路 径	属 主	模 式	内 容
/var/spool/clientmqueue	smmsp	770	初始提交的邮件队列[a]
/var/spool/mqueue	RunAsUser	700	邮件队列目录
/, /var, /var/spool	root	755	到 **mqueue** 的路径
/etc/mail/*	TrustedUser	644	映射、配置文件、别名
/etc/mail	TrustedUser	755	映射文件的父目录
/etc	root	755	到 **Mail** 目录的路径

a. 8.12 及以后版本。

sendmail 不会读取权限过于宽松的文件（例如，任何组或任何人都可写的文件，或者位于任何组或任何人都可写的目录中的文件）。**sendmail** 对所有权和权限很严格的部分原因，是有一些操作系统允许用户用 **chown** 将自己的文件转让出去（那样的操作系统大多都是从 System V 演变而来的）。

Linux 系统默认都有一个健全版本的 **chown**，它们不允许转让文件。不过，在代码中设置一个预编译指令#ifdef 就可以让 **chown** 采用 System V 的语义（CAP_CHOWN）。您还要重新编译内核。但是这种做法并不好，不要迫使您原本合理的 Linux 的 **chown** 按照有缺陷的 System V 的方式来运行。

特别地，**sendmail** 对任何别名（aliases）文件或者转发（forward）文件的完整路径非常挑剔。这么挑剔的话，有时会和站点管理 Majordomo 邮递列表别名的方式发生冲突。例如，如果 Majordomo 列表位于**/usr/local**，那么整个路径必须是可信的，路径的各个部分都没有组写权。这项限制让列表的所有者更难以管理别名文件。要看看在 **sendmail** 对于权限的考虑上，您做得如何，可以运行：

```
# sendmail -v -bi
```

-bi 标志初始化别名数据库，如果权限不对就向您发出警告。

[32] TRUSTED_USERS 功能专门用来支持邮件列表软件。例如，如果要使用 Majordomo，则必须先在 TRUSTED_USERS 类中添加“majordom”用户。root 和 daemon 是这个类的默认用户。

www.TopSage.com

如果到某个文件的目录路径是不安全的（权限过于宽松），**sendmail** 将不再读取链接数大于 1 的**.forward** 文件。这条规则在最近就让 Evi 碰上了，因为她的**.forward** 文件在把她的邮件从她不会收到太多邮件的小站点转发过来时一声不响地就失败了，这个文件在一般情况下是到**.forward.to.boulder** 或**.forward.to.sandiego** 的一个硬链接。过了好几个月她才意识到并明白过来，“我从没收到过您的信”是她的错，并且再也不是一个说得过去的借口了。

可以用 DontBlameSendmail 选项关掉上面提到的许多限制文件访问的规则。但是不要那么做。

18.12.3 发到文件和程序的可靠邮件

我们建议您使用 **smrsh** 代替**/bin/sh** 作为您的邮寄程序（**mailer**），并用 **mail.local** 代替**/bin/mail** 作为本地邮寄程序。这两个程序都包含在 **sendmail** 的软件发布中。为了把它们合并到您的配置中，请把以下两行：

```
FEATURE('smrsh', 'path-to-smrsh')
FEATURE('local_lmtp', 'path-to-mail.local')
```

添加到您的**.mc** 文件中。如果省略了明确的路径，就认为命令位于**/usr/libexec** 下。您可以使用 **sendmail** 的 confEBINDIR 选项把二进制程序的默认位置改为您需要的任何地方。Red Hat 的默认安装根本就没有提供 **mail.local**。SUSE 把它放在**/usr/lib/sendmail.d/bin** 下，而 Debian 和 Ubuntu 把它们藏在了**/usr/lib/sm.bin** 里。

smrsh 是一个受限的 shell，它只执行一个目录中包含的程序（默认情况下是**/usr/adm/sm.bin**）。Red Hat 和 Fedora 把 **smrsh** 这个二进制程序安装在**/usr/sbin** 下，SUSE 把它放在**/usr/lib/sendmail.d/bin** 下，而 Debian 和 Ubuntu 把它放在**/usr/lib/sm.bin** 里。**smrsh** 忽略用户指定的路径，并试图在它自己已知是安全的目录中找到要求执行的任何命令。**smrsh** 还禁止使用某些 shell 元字符，比如“<”，即输入重定向符。在 **sm.bin** 中允许用符号链接，所以您不需要给允许使用的程序做副本[33]。

下面是一些 shell 命令的例子，以及 **smrsh** 给它们可能的解释：

```
vacation eric                  # 执行 /usr/adm/sm.bin/vacation eric
cat /etc/passwd                # 被拒绝，在 sm.bin 里没有 cat 命令
vacation eric < /etc/passwd    # 被拒绝，不允许有<
```

在电子邮件被一个 **aliases** 或者**.forward** 文件重定向到一个文件的时候，**sendmail** 的 SafeFileEnvironment 选项控制着可以把文件写到什么地方。它会让 **sendmail** 执行一次 chroot 系统调用，让文件系统的根不再是/而是**/safe**，或者您在 SafeFileEnvironment 选项中指定的任何路径。例如，把邮件重定向到**/etc/passwd** 文件的别名实际上是写入了**/safe/etc/passwd**。

SafeFileEnvironment 选项也能通过只允许写入普通文件的做法，保护设备文件、目录和其他的特殊文件。除了提高安全性之外，这个选项还有助于缓解用户犯错所造成的影响。有些站点把这个选项设为**/home**，在禁止进入系统文件的时候，能够访问用户的主目录。

邮寄程序（mailer）也可以进入被 **chroot** 过的目录。目前必须在邮寄程序的定义中指定该选项，但它应该很快就能通过 **m4** 来配置了。

18.12.4 隐私选项

sendmail 还具有隐私选项，可控制：

[33] 不要把 **procmail** 这样的能够派生出 shell 的程序放入 **sm.bin**。不要把 **procmail** 用作本地的邮寄程序，因为用户可以从他们的**~/.procmailrc** 文件运行任何他们想要运行的程序。这不安全。

www.TopSage.com

- 哪些外部人员可以由 SMTP 来测定您的站点；
- 您对 SMTP 连接的另一端主机有什么要求；
- 您的用户是否可以看到或者运行邮件队列。

表 18.15 列出了在撰写本书的时候隐私选项可能的值，关于当前的信息请参见软件发布中的 **doc/op/op.ps** 文件。

我们建议您保守一些，请把：

```
define('confPRIVACY_OPTIONS', "goaway, authwarnings, restrictmailq,
        restrictqrun")
```

用在**.mc** 文件中。**sendmail** 隐私选项所用的默认值是 `authwarnings`，上面这一行将重置该值。请注意有两组引号，某些版本的 **m4** 需要用它们把隐私选项值列表中的逗号括起来。Red Hat 和 Fedora 默认选 `authwarnings`，而 SUSE、Debian 和 Ubuntu 默认选 `authwarnings,needmailhelo、novrfy、noexpn` 和 `noverb`。

表 18.15 PrivacyOption 变量的值

值	含 义
Public	不作隐私/安全检查
Needmailhelo	要求有 SMTP HELO（识别远程主机）
Noexpn	不允许 SMTP EXPN 命令
Novrfy	不允许 SMTP VRFY 命令
Needexpnhelo	不扩展没有 HELO 的地址（EXPN）
Needvrfyhelo	不验证没有 HELO 的地址（VRFY）
noverb[a]	不允许 EXPN 使用详细（verbose）模式
Restrictmailq	只允许 **mqueue** 目录的属组看到队列
Restrictqrun	只允许 **mqueue** 目录的属主运行队列
Restrictexpand	用**-bv** 和**-v** 标志限制显示的信息[b]
noetrn[c]	不运行异步方式运行排队操作
Authwarnings	添加 Authentication-Warning（身份验证警告）信头（这是默认值）
Noreceipts	关闭成功回执的 DSN（delivery status notification，投递状态通知）
Nobodyreturn	不在 DSN 中返回消息主体
Goaway	禁止所有的 SMTP 状态查询（EXPN、VRFY 等）

a. 在发出 EXPN 命令的时候，详细模式会根据**.forward** 文件，提供有关用户邮件行踪的更多信息。在任何向外界暴露的机器上，应使用 noverb，当然 noexpn 更好。
b. 除非由 root 或者 `TrustedUser` 执行。
c. ETRN 是一条 ESMTP 命令，它是专为拨号主机的使用而设计的。它要求只对到该主机的消息运行排队操作。

18.12.5 运行 chroot 后的 sendmail

如果您担心 **sendmail** 会访问您的文件系统，那么可以在由 **chroot** 建成的“监管环境（jail）”中启动它。在监管环境中构造一个最小的文件系统，包括**/dev/null**、**/etc** 的要素（**passwd**、**group**、**resolv.conf**、**sendmail.cf**、所有映射文件、**mail/***）、**sendmail** 需要的共享库、**sendmail** 二进制文件、邮件队列目录、以及所有日志文件。您可能不得不手忙脚乱半天才能彻底解决问题。使用 **chroot** 命令启动一个受监控的 **sendmail**。例如：

```
# chroot /jail /usr/sbin/sendmail -bd -q30m
```

www.TopSage.com

18.12.6 拒绝服务攻击

要防止拒绝服务攻击（denial of service，DoS）是很困难的，因为没有一种方法可以预先知道一则消息是一个攻击还是一份有效的电子邮件。攻击者可以尝试各种卑鄙的招数，包括用假连接淹没 SMTP 端口，用巨大的消息填满磁盘分区，阻塞向外的连接，以及邮件炸弹。**sendmail** 有一些配置参数可以帮助减慢或限制一次拒绝服务攻击的冲击，但是这些参数也会妨碍合法的邮件传输。邮件过滤库（milter）可能会帮助系统管理员对抗一次长时间的拒绝服务攻击。

`MaxDaemonChildren` 选项可以限制 **sendmail** 进程的数目。它能防止系统被 **sendmail** 的任务压跨，但也让攻击者能够非常容易地关闭 SMTP 服务。`MaxMessageSize` 选项有助于防止邮件队列目录被填满，但如果把它设置得太小，合法邮件将被弹回（您可以向用户说明这项限制，这样当邮件被弹回时他们就不会感到惊讶。无论如何我们建议把限制设得高一些，因为一些合法的邮件也很大）。`ConnectionRateThrottle` 选项限制的是每秒钟内允许的连接数目，它可以大大减缓连接速度。最后，可以设置 `MaxRcptsPerMessage`，它控制一则消息中允许收件人的最大数目，这可能会有所帮助。

sendmail 一直能够根据系统的平均负载来拒绝链接（`REFUSE_LA` 选项）或者把电子邮件排入队列（`QUEUE_LA` 选项）。在 8.12 版的 sendmail 引入了一种变化（`DELAY_LA` 选项），它能让邮件继续流动，但是速度却被降低了。参考 18.12.3 节关于性能的详细说明。

尽管这一切都致力于保护您的邮件系统，但用邮件轰炸您的人还是会妨碍合法的邮件传输。邮件炸弹是相当卑鄙的。

科罗拉多大学为每个学生（大约有 25000 人）提供了一个电子邮件账户，**pine** 是默认的邮件阅读器。几年前，一位学生在本地的计算机商店获得了一份新工作，他的雇主说服他提供口令文件的一个副本。然后这家公司给口令文件中的每个人都发送了一则广告，大约分成每次 1000 个收件人（这使得 To: 行非常长）这样来发。

pine 是把默认的回复模式设置为对所有的收件人和发件人作出答复来编译的。许多学生用这样的问题来做答复："为什么您给我发送这封垃圾？"。当然这样的信发给了 To: 行中的其他所有人。结果是服务器拒绝所有服务——无论是电子邮件还是任何其他使用。**sendmail** 接管了所有的 CPU 计算周期，邮件队列庞大无比，所有正在进行的工作都被暂停。唯一的解决方案是让这台机器下线，进入每个用户的邮件缓冲区，并删除讨厌的消息（也已经用了对标题[Subject]行进行信头检查的方法）。

18.12.7 伪造

伪造电子邮件在过去是小事一桩。在 8.10 以前，任何用户都可以伪造邮件，使得它看上去就像出自您所在的域。从 **sendmail** 8.10 开始包含 SMTP 身份验证，可以验证发送方机器的身份。必须用 AuthMechanisms 选项打开身份检查功能。遗憾的是，**sendmail** 的身份验证功能不是端到端的，而只是在相邻的服务器之间。如果一则消息是由几台服务器处理的，那么虽然身份验证有帮助，但却不能保证不会出现伪造的消息。

类似地，在邮件消息中模仿任何用户也同样是可能的。如果邮件消息是您所在单位用于钥匙、访问卡和金钱等事情的权威传达媒介，那就要小心了。您应该用这一事实警告负责管理的用户，并建议如果他们见到的可疑邮件似乎出自某个权威人士，就应该验证该消息的有效性。如果这则消息要求把不合理的特权授予某个不寻常的人，就更要加倍小心了。如果邮件把重要的钥匙给了一个尚未毕业的大学生，这就很可疑！

`authwarnings` 隐私选项可以标记出本地的伪造企图，它给看上去像是伪造的发出邮件添加

www.TopSage.com

一个 Authentication-Warning 信头。不过，许多用户代理程序在默认情况下会隐藏这个信头。

如果伪造的邮件出自您所控制的机器，那么您完全可以做点儿工作来阻止它。可以使用 **identd** 守护进程验证发件人的真实登录名。**sendmail** 回叫发送的主机，要求在那台主机上运行的 **identd** 查找那个发送邮件的用户的登录名。如果远程主机上没有运行 **identd**，**sendmail** 将一无所获。如果远程机器是一台单用户的工作站，其拥有者就可以把 **identd** 配置成返回虚假的回答。但是如果远程主机是一台多用户机器，比如许多大学的计算中心建立的那种机器，**identd** 就会返回该用户的真实登录名，以便 **sendmail** 把它插入到消息的信头中。

许多站点不会运行 **identd**，它也常常被防火墙所阻挡。**identd** 只在网点内部才真正有用，因为在您控制之外的机器可以说谎。在有某些不负责任的用户的大型网点（例如，某所大学）上，这种方法效果非常好——但是也会降低 **sendmail** 的性能。

几年前，当我们第一次试用 **identd** 时，我们站点上的一个学生对他所在高级项目组的成员感到灰心失望。他企图扮成自己的指导教师给他的队友发邮件，告诉他们他知道他们没有尽心尽力，应该工作得更卖力些。很遗憾，他犯了个语法错误，这则消息被弹回给了指导教师。**sendmail** 使用的 IDENT 协议告诉了我们他是谁。**sendmail** 在弹回的消息中包含如下几行：

```
The original message was received at Wed, 9 Mar 1994 14:51 -0700 from
    student@benji.Colorado.EDU [128.138.126.10]
```

但是消息信头本身却说的另一番故事：

```
From: instructor@cs.Colorado.EDU
```

教训：在偷偷摸摸干坏事时要避免语法错误。根据我们针对伪造邮件的政策，这个学生的登录名在那个学期的余下时间中被禁用，这刚好满足了他的愿望。他不能继续干那个项目了，他的伙伴们不得不打起精神好好工作。

18.12.8 消息的隐私

除非您使用外部加密软件包，比如 PGP（Pretty Good Privacy）或者 S/MIME，否则基本上是保证不了消息隐私的。默认情况下，所有邮件在发送时都不加密。端到端的加密要求得到邮件用户代理的支持才行。有关 PGP 的更多信息请参见 20.11.2 节。

S/MIMI 和 PGP 都是在 RFC 的系列文档中说明的，S/MIME 则处于标准系列。不过，我们更偏向 PGP，它的使用范围更广，而且是由一位优秀的密码学专家 Phil Zimmermann 设计的，我们都信任他。这些行将出现的标准为邮件的机密性、身份验证、保证消息的完整性、以及发件来源的认可方面提供了一个基础。因为信头和信封仍然是以明文来发送的，所以还可以进行流量分析。

告诉您的用户，如果他们想要让自己的邮件保密，就必须自行加密。

18.12.9 SASL：简单的身份验证和安全层

sendmail 8.10 和后继版本都支持 RFC2554 中定义的 SMTP 身份验证。它基于 SASL（Simple Authentication and Security Layer，简单的身份验证和加密协议层）。SASL 是一种共享秘密的系统，一般用于主机对主机的验证，您必须在彼此互相要验证对方身份的每一对服务器上明确地进行配置。

SASL 是一种通用的身份验证机制，可以集成到各种协议中去。**sendmail**、Cyrus 的 **imapd**、Netscape、Outlook、Thunderbird 和 Eudora 的某些版本都可以用它。SASL 框架（它是一个库）有两个基本概念：一个授权标识符和一个身份验证标识符。它可以把这些标识符映射到文件权限、账号口令、

www.TopSage.com

Kerberos tickets 等上面。SASL 同时包含身份验证组件和加密组件。要配合 **sendmail** 使用 SASL 的话，可以从 asg.web.cmu.edu/sasl 获得 Cyrus SASL。

TLS 是在 RFC2487 中制定的另一种加密/验证系统。它在 **sendmail** 中是作为一种对 SMTP 的扩展来实现的，这叫做 STARTTLS。甚至您可以 SASL 和 TLS 两者都用。

TLS 设置起来要稍微困难的一点儿，它要求有证书服务。您可以向 VeriSign 支付一大笔钱来发布您自己的证书（标识一个实体的签名公钥），或者建立一个您自己的证书服务。它用更强的验证手段代替了主机名或者 IP 地址，作为授权中转邮件主机或者接受主机链接的标记。在 access_db 中像下面这样的配置项：

```
TLS_Srv:secure.example.com    ENCR:112
TLS_Clt:laptop.example.com    PERM+VERIFY:112
```

指出要使用 STARTTLS，发送到 secure.example.com 这个域的邮件都必须用至少 112 位的密钥进行加密。从 laptop.example.com 这个域内的主机发来的邮件只有客户机本身通过身份验证才能接受。

Sendmail，Inc.公司的另一位员工 Gregory Shapiro 编写了一些有关安全和 **sendmail** 的教程，它们很不错，从 www.sendmail.org/~gshapiro 可以得到它们。

18.13 sendmail 的性能

sendmail 有几个配置选项是专为改善性能而设计的。虽然我们已经在本章中间分散介绍了它们，但是我们还是要在这一节对最重要的几项展开来阐述一下。如果您所运行的是个容量很大的邮件系统（无论哪一个方向），那么都应该考虑这些选项和功能。实际上，如果您真的需要在一小时内发送 100 万封邮件，而您又不是在发垃圾邮件，那么您最好求助于 **sendmail** 的商业一面，即 Sendmail，Inc.。

18.13.1 投递方式

sendmail 有 4 种基本的投递方式：后台（background）、交互（interactive）、队列（queue）和推迟（defer）。每种方式都代表着在延迟和吞吐量之间的一种折衷。后台方式会立即投递邮件，但要求 **sendmail** 派生出一个新的进程来完成投递。交互方式也立即投递邮件，但是由同一个进程完成投递，而且要让远端等待结果。队列方式把传入的邮件排入队列，在稍晚些的时候由一个队列运行器（queue runner）来投递。推迟方式和队列方式类似，但是它推迟映射、DNS、别名和转发的查找操作。交互方式很少用。后台方式延迟最低，而推迟或者队列方式吞吐量较高。投递方式是用 confDELIVERY_MODE 这个选项设置的，它默认为后台方式。

18.13.2 队列分组和信封分割

队列组（queue group）是在 **sendmail** 8.12 里出现的新功能，它能让您为传入邮件创建多个队列，并且分别控制每个队列组的属性。队列组包括一个或者几个队列目录。例如，如果您的 Linux 主机是一个 ISP 的邮件枢纽主机，您可能会给拨号用户定义一个队列组，而且允许拨号用户在他们连接上来下载自己的邮件时，发起一次队列运行操作（使用 SMTP 命令 ETRN）。队列组可以配合信封分割（envelope-splitting）的功能使用，后者可以把带有多个收件人的信封划分到多个队列组中。这一功能加上为每个队列组使用多个队列目录，就能有助于缓解在一个文件系统中有太多文件所造成的性能问题[34]。

[34] 如果您在用一个 ext3 文件系统的同时也在使用 2.6 以上版本的内核，那么目录索引也可能有助于减轻大目录造成的性能影响。您可以用命令 **tune2fs-O dir_index** 把一个已有的文件系统改为使用这个特性。

www.TopSage.com

当一条消息进入邮件系统的时候，它就分到了一个或者几个队列组中。每个收件人的队列组是独立决定的。信封要根据队列组的分配情况重新写。如果使用多个队列目录，那么消息会随机地分配到正确队列组的若干个队列中。

如果一个队列组对每个信封上收件人的最大数目有限制，那么 **sendmail** 就把消息的信封分割为几个能够符合队列组的参数小信封。

队列组是用**.mc** 文件里的指令声明的，但却是由 LOCAL_RULESETS 选项在原始配置文件里配置的，我们根本没有在本书里介绍过这个选项。如果您想要使用队列组来改善性能，或者向不同的目的地提供不同的服务质量，那么就可以用下面的例子作为起点。

表 18.16 列举了可以为一个队列组指定的属性。在定义队列组的时候，只需要指定属性名字的第一个字母。

表 18.16 队列组的属性

属 性	含 义
Flags	多用于未来的控制功能；必须设置 f 标志才能有多个队列运行器
Nice	这个队列组的优先级；优先级较低，则以 **nice** 系统调用的方式运行
Interval	在队列运行之间等待的时间
Path	和队列组相关联的队列目录的路径（必须要有）
Runners	在队列组上并发运行的 **sendmail** 进程数目
Recipients	每个信封上最大的收件人数目

下面是一个例子，它有针对本地邮件、发送到 aol.com 的邮件、发送到其他远程站点的邮件、以及其余所有邮件的默认队列的队列组。下面这几行将会放到**.mc** 文件的正规部分里面：

```
dnl ##### -- queues
QUEUE_GROUP(`local', `P=/var/spool/mqueue/local')
QUEUE_GROUP(`aol', `P=/var/spool/mqueue/aol, F=f, r=100')
QUEUE_GROUP(`remote', `P=/var/spool/mqueue/remote, F=f')
```

然后在**.mc** 文件的末尾加上：

```
LOCAL_RULESETS
Squeuegroup
R<$+>                $1
R$*@aol.com          $# aol
R$*@mydomain.com     $# local
R$*@$*               $# remote
R$*                  $# mqueue
```

在本例中，当我们定义 AOL 的队列组时，我们特别限制每个消息最多有 100 个收件人。如果一则传出的消息有 10 000 个收件人，其中 1 234 个收件人是在 AOL，那么就会分割信封，在 aol 队列组中放入 13 条消息，其中 12 条消息有 100 个收件人，而 1 条消息有剩下的 34 个收件人。

为了提高速度，可以尝试采用快速分割，它会在分类处理期间推迟做 MX 查询：

```
define('confFAST_SPLIT', '1')
```

18.13.3 队列运行器

sendmail 派生出自己的若干个副本执行实际的邮件传输操作。您可以控制在任何指定的时间，可以有多少个 **sendmail** 的副本在同时运行，甚至还可以控制每个队列组上关联多少个 **sendmail** 的副本。

www.TopSage.com

这个功能能够让系统管理员在他们繁忙的邮件枢纽主机上达到 **sendmail** 和操作系统之间的平衡。

sendmail 的 3 个选项控制着每个队列上队列运行器守护进程的数目。

- MAX_DAEMON_CHILDREN 选项指定了在任何时刻允许运行的 **sendmail** 守护进程的副本数量，其中包括那些正在队列上运行的以及那些正在接受传入邮件的副本。
- MAX_QUEUE_CHILDREN 选项设置一次所能允许的队列运行器的最大数目。
- 如果在队列组的定义中没有用 Runners=（或者 R=）明确设置一个值的话，那么 MAX_RUNNERS_PER_QUEUE 选项设置每个队列上默认的运行器数的上限。

如果您设置的值有冲突（例如，总共最多有 50 个队列运行器，但是 10 个用于本地队列、30 个用于 mydomain 队列、还有 50 个用于 AOL 队列），那么 **sendmail** 会把队列分成工作组（workgroup），在工作组间进行循环。在本例中，本地和 mydomain 队列都在一个工作组里，而 AOL 队列在第二个工作组里。如果您选择的限制必须发生冲突（例如，max=50 但是 AOL=100），那么 **sendmail** 会使用 MAX_QUEUE_CHILDREN 作为它的队列运行器数量的绝对上限。

18.13.4 负载均衡控制

在系统平均负载太大的时候，**sendmail** 一直能够拒绝连接或者把消息排入队列，而不会投递它们。遗憾的是，平均负载的时间粒度只是一分钟，所以它不是一个能精确刻画 **sendmail** 所占资源的工具。新的 DELAY_LA 原语可以让您设定一个负载平均值，在达到这个值的时候，**sendmail** 就会放慢执行的速度，它在为当前连接执行 SMTP 命令之后到接受新的连接之前要睡上一秒钟。默认值是 0，意味着关闭这一机制。

18.13.5 队列中无法投递的消息

在繁忙的邮件服务器上，在邮件队列中无法投递的消息实际上是性能杀手。**sendmail** 有几项功能能够帮助解决消息无法投递造成的问题。最有效果的就是 FALLBACK_MX 选项，它把第一次尝试无法投递出去的消息转给另一台机器。这项功能可以让您的主邮件服务器选择发送地址正确的消息，而把问题留给第二台后援（fallback）主机。另一项有帮助的功能是主机状态目录，它保存有在运行队列上的远程主机的状态。

对于有大型邮递列表的站点来说，由于这些站点不可避免会包含许多暂时或者永远无法投递的地址，采用 FALLBACK_MX 选项就可以在性能上取得很大收效。要使用这个选项，您必须为推迟处理的邮件指定后援主机。例如，

```
define('confFALLBACK_MX', 'mailbackup.atrust.com')
```

会把所有第一次投递失败的消息转发给中央服务器 mailbackup.xor.com，由它做进一步的处理。在 8.12 版的 **sendmail** 上，如果在 DNS 中给多台指定的主机设定了多条 MX 记录，那么就可以有多台后援主机。

在后援机器上，您可以使用 HOST_STATUS_DIRECTORY 选项帮助处理多起投递失败的消息。这个选项指导 **sendmail** 给每台接受邮件的后援主机维护一个状态文件，并且在每次运行队列的时候使用那个状态信息确定该主机的优先级。这个状态信息有效地实现对失败结果的高速缓存，并且可以在多个运行的队列上共享这一信息。在处理带有许多错误地址的邮递列表的主机上，这项功能会在性能上取得成效，但是在文件 I/O 方面的开销相当大。

下面是一个例子，它使用目录**/var/spool/mqueue/.hoststat**（要先创建这个目录）：

```
define('confHOST_STATUS_DIRECTORY', '/var/spool/mqueue/.hoststat')
```

www.TopSage.com

如果采用相对路径指定**.hoststat** 目录，那么它就会保存在队列目录下面。**sendmail** 根据目的主机名创建它自己内部的子目录层次结构。

例如，如果邮件无法发送到 evi@anchor.cs.colorado.edu，那么状态信息就进入到目录**/var/spool/mqueue/.hoststat/edu./colorado./cs./**下的一个叫做 **anchor** 的文件里，因为主机 **anchor** 就是它自己的优先级最高的 MX 记录。如果 DNS MX 把 anchor 的邮件指向了主机 foo，那么文件名就会变成 **foo**，而不是 **anchor**。

繁忙主机所能采用的第三项改善性能的措施是把排队的时间设定到一个最小值，让首次投递失败的任何消息都被排入队列，在队列中发送它的尝试时间间隔最短。这项技术通常要配合 **sendmail** 频繁运行 **sendmail** 的命令行标志（例如，**-q5m**）一起使用。如果一个队列运行器在一则消息上被耽搁了，那么另一个队列运行器就会在 5 分钟内启动起来，从而提高了投递消息的性能。整个队列根据哪些消息已经到了所要求的最短时间来成批运行。带上标志**-bd -q5m** 运行 **sendmail** 时，把下面这个选项：

```
define('confMIN-QUEUE_AGE', '27m')
```

包括在配置文件中可以带来响应速度更快的系统。

18.13.6 内核调优

如果您计划用一台 Linux 机器作为大容量的邮件服务器，那么应该修改内核连网配置参数中的几个暗示，或许还要构造一个定制的内核（取决于您的硬件配置和预计的负载）。删掉任何不必要的驱动程序，就可以从一个改进后的内核启动，这个内核只适用于您自己的硬件配置。

如果主机有一个以上的处理器的时候，定制的内核应该包括对多处理器（SMP）的支持（我们认识到，对于真正的 Linux 迷来说，这条建议可以比作是“不要忘记喘气”这样几乎不需要提醒的提醒。但是，既然 Linux 用户在构造内核上的技能参差不齐，所以我们就没有理会审稿人的意见，把这段提醒的话保留了下来）。

要重新设置连网协议栈的参数，需使用 shell 的 **echo** 命令重新设置**/proc** 文件系统中的适当变量。第 12 章里介绍了这个过程总体性的知识。表 18.17 给出了在高容量的邮件服务器上要改的参数，以及它们的建议和默认值。这些更改操作或许应该放在一个 shell 脚本里，在系统引导的时刻运行，执行相应的 **echo** 命令。

表 18.17 在大容量的邮件服务器上需改动的内核参数

变量（相对于/proc/sys）	默 认 值	建 议 值
net/ipv4/tcp_fin_timeout	180	30
net/ipv4/tcp_keepalive_time	7 200	1 800
net/core/netdev_max_backlog	300	1 024
fs/file_max	4 096	16 384
fs/inode_max	16 384	65 536

例如，您可以使用命令：

```
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
```

改变 TCP 的 FIN 超时值。

18.14 sendmail 的统计信息、测试和调试

sendmail 可以收集有关它所处理消息的数量和大小的统计信息。您可以用 **mailstats** 命令显示这些

www.TopSage.com

数据，数据是按邮寄程序（mailer）进行组织的。**sendmail** 的 confSTATUS_FILE 选项（在 OSTYPE 文件中）指定了应该保存统计信息的文件名称。如果指定的文件存在，则将打开记账（accounting）功能。

sendmail 统计信息文件的默认位置是**/etc/mail/statistics**，但有些厂商把这个文件叫做 **sendmail.st**，放在**/var/log** 中。**mailstats** 给出的总计值是从创建统计信息文件以来的累计。如果希望周期的统计信息，可以用 **cron** 轮换并重新初始化此文件。

下面是一个例子：

```
$ mailstats
Statistics from Tue Aug  1 02:13:30 2006
 M   msgsfr  bytes_from  msgsto  bytes_to  msgsrej  msgsdis  Mailer
 4      12        25K      63      455K        0        0  esmtp
 7       0         0K      18       25K        0        0  relay
 8      54       472K       0        0K        0        0  local
=============================================================
 T      66       497K      81      480K        0        0
 C      66                 81                0
```

如果所有人都可以读取邮件统计文件，那么就不必由 root 来运行 **mailstats** 命令。

这条命令显示了 6 个值：收到的消息和收到的千字节数（msgsfr、bytes_from），发送的消息和发送的千字节数(msgsto、bytes_to)，拒收的消息数(msgsrej)，以及丢弃的消息数(msgsdis)。第一列的数字标识出邮寄程序（mailer），最后一列列出了邮寄程序的名字。T 开头的行是所有的消息和字节数，C 开头的行是连接数。这些值中包括了本地和中继转发的邮件。

18.14.1 测试和调试

基于 **m4** 的配置要经过某种程度上的预先测试。如果要用它们，可能并不需要进行底层的调试。有一样东西是调试标志不能测试的，那就是您的设计方案。在研究本章时，我们在所检查的好几个配置文件和设计中发现了错误。这些错误小到调用某项功能时没带上必要的宏（例如，使用 masquerade_envelope 时没有用 MASQUERADE_AS 打开伪装），大到 **sendmail** 配置和防火墙的设计之间发生全面冲突，而防火墙控制着是否允许邮件进入以及进入的条件有哪些。

您不能设计一个在真空中的邮件系统。邮件系统必须与 DNS MX 记录和防火墙策略同步（或者至少不会相冲突）。

sendmail 提供了世界上已知最丰富的调试辅助程序集之一，它的调试标志不是简单的布尔值或整数，而是二维量 *x.y*，其中 *x* 选定主题，而 *y* 选定要显示的信息量。值为 0 表示不调试，为 127 表示如果您打印输出的话，会造成纸张的浪费。主题的范围是 0~99，目前定义了大约 80 个。

sendmail 软件发布中的文件 **sendmail/TRACEFLAGS** 列出了正在使用的值和使用它们的文件和函数。所有对调试的支持都处在原始配置文件的层次上。在许多情况下，把调试输出和 **sendmail** 源代码一起看看是会有帮助的。

如果调用 **sendmail** 时带上一个**-d***x.y* 标志，调试输出将送到屏幕（标准错误输出）。表 18.18 显示了几个重要的 *x* 值和一些 *y* 的建议值。如果您打开了作为守护进程（**-bd**）运行的 **sendmail** 的调试功能，那么就要小心了，因为调试输出可能最终会卡住 SMTP 对话，在 **sendmail** 和远程主机进行通信时造成奇怪的故障。

表 18.18　　调试主题

主　题	含义和建议
0	显示编译标志和系统身份（请试试 y = 1 或者 10）
8	显示 DNS 名字解析（试试 y=8）

www.TopSage.com

续表

主 题	含义和建议
11	跟踪邮件的投递（显示邮寄程序的调用）
12	显示本地到远程的名字转换过程
17	列出 MX 主机
21	跟踪重写规则（要了解详情，可以使用 y=2 或者 y=12）
27	显示别名和转发机制（试试 y=4）
44	显示如果发生故障，尝试打开文件的次数（y=4）
60	显示数据库映射的查找

Gene Kim 和 Rob Kolstad 写过一个叫 **checksendmail** 的 Perl 脚本，它用地址测试模式调用 **sendmail**，在包含由您提供的测试地址的文件上运行。它将结果与期望值相比较。这个脚本让您能够根据站点上典型地址的测试集，对新版的配置文件进行测试，确保自己不会在无意间破坏以往一直工作正常的任何地方。

18.14.2 啰嗦的投递

许多在命令行上调用 **sendmail** 的用户代理程序都能接受一个**-v** 标志，它传递给了 **sendmail**，让 **sendmail** 显示出投递消息所采取的步骤。下面的例子使用了 **mail** 命令。粗体字是从键盘向用户代理输入的内容，其余的内容是 **sendmail** 的啰嗦输出。

```
$ mail -v trent@toadranch.com
Subject: just testing, please ignore
hi
.
Cc:
trent@toadranch.com... Connecting to coyote.toadranch.com. via esmtp...
220 coyote.toadranch.com ESMTP Sendmail 8.11.0/8.11.0; Tue, 7 Aug 2001 20:
    08:51 -0600
>>> EHLO anchor.cs.colorado.edu
250-coyote.toadranch.com Hello anchor.cs.colorado.edu [128.138.242.1], pleased
    to meet you
250-ENHANCEDSTATUSCODES
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250-AUTH DIGEST-MD5 CRAM-MD5
250 HELP
>>> MAIL From:<evi@anchor.cs.colorado.edu> SIZE=65
250 2.1.0 <evi@anchor.cs.colorado.edu>... Sender ok
>>> RCPT To:<trent@toadranch.com>
250 2.1.5 <trent@toadranch.com>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
```

www.TopSage.com

```
250 2.0.0 f7828pi03229 Message accepted for delivery
trent@toadranch.com... Sent (f7828pi03229 Message accepted for delivery)
Closing connection to coyote.toadranch.com.
>>> QUIT
221 2.0.0 coyote.toadranch.com closing connection
```

anchor 上的 **sendmail** 连接上了 toadranch.com 上的 **sendmail**。两台机器都使用 ESMTP 协议来协商交换消息。

18.14.3 用 SMTP 通信

在调试邮件系统时可以直接使用 SMTP。为了发起一次 SMTP 会话，可以 **telnet** 至 TCP 端口 25 或者使用 **Sendmail-bs** 命令。默认情况下，这是 **sendmail** 在守护进程模式（**-bd**）下运行时进行监听的端口，它以邮件提交代理身份运行时使用端口 587。表 18.19 显示了最重要的 SMTP 命令。

表 18.19 SMTP 命令

命 令	函 数
HELO *hostname*	如果采用 SMTP 通信，标识正在连接的主机
EHLO *hostname*	如果采用 ESMTP 通信，标识正在连接的主机
MAIL From: *revpath*	发起一次邮件事务（信封发件人）
RCPT To: *fwdpath*[a]	标识信封收件人
VRFY *address*	验证地址是否有效（可投递）
EXPN *address*	显示别名和**.forward** 映射的展开结果
DATA	开始消息的主体[b]
QUIT	结束交换并关闭连接
RSET	重置连接状态
HELP	打印 SMTP 命令的汇总信息

a. 一则消息可以用多条 RCPT 命令。
b. 您可以在消息主体内的一行上输入一个点（.）来结束消息主体。

SMTP 整个语言只有 14 条命令，所以它非常容易学习和使用。它不区分大小写。SMTP 的规范可以在 RFC2821 中找到。

包括 **sendmail** 在内的大多数传输代理都支持 SMTP 和 ESMTP，**smap** 是现如今唯一的例外。遗憾的是，许多提供主动过滤机制的防火墙设备并不支持 ESMTP。

支持 ESMTP 协议的进程将以 EHLO 命令而不是 HELO 发起通信。如果另一端的进程听懂了并答复 OK，那么双方便协商它们能支持的扩展，最后达到进行交流所需的最小公分母。如果返回的是一条错误消息，那么支持 ESMTP 协议的进程就后退到 SMTP。

18.14.4 队列监视

您可以使用 **mailq** 命令（等价于 **sendmail -bp** 命令）查看队列内消息的状态。消息在被投递时，或者已经尝试过投递但没有成功的时候，它们就被排入“队列”。

mailq 打印出当时**/var/spool/mqueue** 下文件的汇总信息，这些信息普通人就可以直接阅读。它的输出对于判断一则消息为什么被拖延非常有帮助。如果信息显示出正在形成积压邮件的现象，那么您就可以监视 sendmail 尝试消除卡壳的状态。

注意，**sendmail** 在 8.12 及以后的版本上，有两个默认的队列：一个用于端口 25 上接收到的

www.TopSage.com

消息，另一个用于在端口 587 上接收到的消息（客户机提交队列）。您可以调用 **mailq-Ac** 以看到客户机队列。

下面是 **mailq** 的一种典型输出。这个例子显示出有 3 条消息在等候投递：

```
$ sudo mailq
/var/spool/mqueue (3 requests)
-----Q-ID-----      --Size-- -----Q-Time-----  ------------Sender/Recipient-----------
k623gYYk008732   23217  Sat Jul 1 21:42  MAILER-DAEMON
     8BITMIME  (Deferred: Connection refused by agribusinessonline.com.)
                                            <Nimtz@agribusinessonline.com>
k5ULkAHB032374  279     Fri Jun 30 15:46  <randy@atrust.com>
             (Deferred: Name server: k2wireless.com.: host name lookup fa)
                                            <relder@k2wireless.com>
k5UJDm72023576  2485    Fri Jun 30 13:13  MAILER-DAEMON
             (reply: read error from mx4.level3.com.)
                                            <lfinist@bbnplanet.com>
```

如果您认为自己比 **sendmail** 还了解情况，或者您只是想要 **sendmail** 尝试立即重新投递排队的消息，那么可以运行命令 **sendmail -q** 迫使队列开始运行。如果您使用 **sendmail -q -v** 命令，那么 **sendmail** 会一次一次地显示每次投递的情况，这往往能够帮助进行调试工作。如果让 **sendmail** 自己自行处理的话，它会每隔一段时间（一般是 30 分钟）重新尝试一次投递。

18.14.5 日志机制

sendmail 使用 syslog（系统日志）来记录错误和状态消息，它所采用的系统日志设备是“mail”，级别可以从“debug”到“crit”，这些消息都以字符串“sendmail”作标记。可以用命令行的选项**-L** 覆盖这个日志字符串“sendmail”，如果您正在调试 **sendmail** 的一个副本，而其他副本正在完成正常的邮件处理事务时，采取这个办法就很方便了。有关系统日志的更多信息请参见第 10 章。

在命令行或配置文件中指定 confLOG_LEVEL 选项，就能决定 **sendmail** 用作日志阈值的严重性级别。日志级别的值越高，表示严重性级别越低，也使得被记录的信息越多。

回想一下，系统日志中记录某个级别的消息，将要报告给该级别以及所有高于它的级别。**/etc/syslog.conf** 文件决定了每则消息最终的目的地。

在 Red Hat 和 Fedora 系统上，**sendmail** 的日志默认记录在**/var/log/maillog** 里。对于 SUSE 来说，这个日志文件是**/var/log/mail**，而对于 Debian 和 Ubuntu 而言，它是**/var/log/mail.log**。如果 Linux 的标准化工作能够整顿这些随机而且显然没有意义的差异，那么我们脚本的可移植性不就会更好吗？

表 18.20 给出了 **sendmail** 日志级别和 syslog（系统日志）严重性级别之间大致的对照关系。

表 18.20　　sendmail 日志级别和 syslog 日志级别的对照

L	级　别	L	级　别
0	不作日志	4	notice
1	alert 或者 crit	5~11	info
2	crit	>=12	debug
3	err 或者 warning		

一个更好的程序是 **mreport**，它是由 Jason Armstrong 编写的。

它可以用来汇总由 **sendmail** 写下来的日志文件。只要运行 **make**，然后再运行 **make install** 立即就能编好它。下面是从一个 Red Hat 系统上取得的 **mreport** 输出结果的样本：

www.TopSage.com

```
# mreport -f -i /var/log/maillog -o mreport.out
[redhat.toadranch.com] [/var/log/maillog]

* [  7]  592601  herb@yoherb.com                trent@toadranch.com
* [  8]  505797  SNYDERGA@simon.rochester.edu   trent@toadranch.com
  [  1]  179386  steph@toadranch.com            bennettr@ci.boulder.co.us
  [  1]   65086  herb@yoherb.com                ned@xor.co
  [  7]   19029  evi@anchor.cs.colorado.edu     trent@toadranch.com
  [ 11]   17677  lunch-request@moose.org        trent@toadranch.com
  [  2]   16178  trent@toadranch.com            ned@camelspit.org
  [  3]   15229  reminders@yahoo-inc.com        herb@toadranch.com
  [  2]    4653  trent@toadranch.com            garth@cs.colorado.edu
  [  2]    1816  UNKNOWN                        trent@toadranch.com

. . . many lines deleted . . .

======================
Total Bytes          : 7876372
Number of Records    : 192
----------------------
User Unknown         : 125
----------------------
Host Name            : redhat.toadranch.com
Input File           : maillog
Output File          : mreport.out
First Record         : Aug  5 04:47:31
Last Record          : Aug  7 18:16:25
----------------------
Time Taken           :  24317 µs
======================
```

如果您的邮件日志文件只能由 root 读取（它们应该如此），您必须使用 **sudo** 或者成为 root 来运行 **mreport**。**mreport** 的标志和选项都在它的手册页中有文档说明。在本例中的**-f** 标志表示根据发件人归并和分类；类似的，**-t** 标志则表示根据收件人进行分类。**-i** 参数是输入文件，**-o** 是输出文件。

如果打算定期运行 **mreport**，那么应该告诉您的用户。他们可能会觉得让系统管理员用这样一种好工具来浏览邮件日志是对他们隐私的一种侵犯。

18.15 Exim 邮件系统

Debian 和 Ubuntu 用 Exim 邮件传输代理取代了 **sendmail**。它是一种组成更简单的系统，但对于邮件负载不重，而且没有特别的邮件系统设计的站点来说，它的特点就是配置更容易。Exim 的功能可以和 **sendmail** 最常用的功能相对应。

18.15.1 历史

Exim 是由剑桥大学（University of Cambridge）的 Philip Hazel 在 1995 年编写的，它按照 GNU 公共许可证来发布。Exim 第 4 版（也叫做 exim4）于 2002 年发布，它现在是 Debian 和 Ubuntu 发布的版本。我们根据 www.exim.org 上的 Exim 文档了解了有关 Exim 的详细配置，并且在这里只简要介绍一下它。

18.15.2 Linux 上的 Exim

和 **sendmail** 软件包里的情况一样，Exim 里也有一些单独的命令来执行特定的邮件功能。这些命

令实际上都是通过以某种命令行标志调用 **exim** 来实现的。表 18.21 列出了这些功能及其对应的 exim 标志。

表 18.21 exim 的工具程序（以及对应的 exim 标志）

命 令	标 志	功 能
mailq	**-bp**	显示邮件队列
rsmtp	**-bS**	批处理 SMTP 连接
rmail	**-i**	和 **smail** 保持兼容[a]
runq	**-q**	运行邮件队列

a. 接受以消息里一行上的点（.）来结束的消息。

18.15.3 配置 Exim

Exim 包含 3 个逻辑上的组成部分：指挥（director）、路由（router）和传输（transport）。指挥部分处理本地地址——也就是说，在本地域内的地址。路由部分处理远程地址，传输部分执行实际的投递操作。

Exim 配置起来很像 **smail3** 或者 Postfix，它也采用形如 *keyword = value* 这样的配置语言。到了 Exim4，系统通过**/etc/exim4/conf.d** 下的一组精巧的配置文件来配置。在您加入或者修改了其中的一个配置文件之后，您必须运行 **update-exim4.conf** 这个脚本，它会创建**/var/lib/exim4/config.autogenerated**。这个自动生成的文件供 Exim 运行的时候使用。

幸好 **Exim** 配置文件的文档说明写得好，每个变量的前面都有注释，介绍这个变量能做什么、常用的值是什么、以及如果您没有定义它可能出现什么样的（糟糕）问题。默认的配置文件定义了大约 100 个变量，但这是一个很大的样板，列举了前面的章节里介绍的 **sendmail** 的很多功能。

在我们作为测试床的系统上，我们不得不设置 4 个变量来启动基本的邮件功能。实际上，那些变量里有两个和隐私问题有关，并不一定需要。下面是这 4 个变量，我们在旁边写了注释：

```
qualify_domain = domain-name      ### 默认设置为主机名(非完整域名)
local_domains = localhost:domain-name
smtp_verify = false               ### 默认为 on,off 禁用 SMTP VRFY
modemask = 002                    ### 默认值为 022,每个用户一个组
```

下面，我们从 **sendmail** 的角度介绍 Exim 的几项功能，以便读者能够比较这两种系统的功能，并且决定自己是否想安装真正的 **sendmail** 来代替 Exim。

18.15.4 Exim 和 sendmail 的相似之处

exim 就像 **sendmail** 以前（8.12 之前）那样，以 setuid 方式成为 root 来运行，所以您需要注意打上安全补丁保持更新。Exim 的可信用户概念和 **sendmail** 的相符，主要用来方便邮递列表的管理，邮递列表的 From 行一般都要重写。Exim 也能让您定义作为系统管理员的用户，他们可以启动队列运行。它默认也允许 SMTP 的验证命令（VRFY）。

当改动了 **exim** 的配置之后，必须用 **kill** 命令向 **exim** 发送 SIGHUP 信号。**exim** 把它的进程 ID 保存在文件**/var/spool/exim/exim-daemon.pid** 里面。它一般把日志记录到**/var/log/exim** 目录下的文件里，但是也可以用 **syslog**。在默认情况下，**exim** 会把日志记录到它自己的文件中，而忽略 syslog，虽然 **syslog.conf** 配置了几个邮件日志文件，但它们都是空的，而且每天也会轮换和压缩（空文件在压缩后比压缩前更大!）。

Exim 允许您把传出的邮件转发给一个聪明的主机，而且允许您在主机和用户两个层面上过滤传

www.TopSage.com

入的邮件。它支持虚拟域，而且有一个重试数据库，在功能上类似于 **sendmail** 为跟踪难以投递的邮件而保持的主机状态目录。**Exim** 有一个系统范围的（不是逐个用户的）过滤机制，能够过滤微软文件的附件、蠕虫或者病毒等。

Exim 包括一项 **sendmail** 所没有的好特性，也就是能够识别出别名和转发的循环，并且合理地处理它们。

最后，**man -k exim** 会举出几种有用的工具，能帮助保持邮件系统的有条不紊。

18.16 Postfix 邮件系统

Postfix 是另一种 **sendmail** 的替代软件。Wietse Venema 到 IBM 的 T. J. Watson 研究中心（T. J. Watson Research Center）做学术休假的一年里，启动了 Postfix 这个项目。Postfix 的设计目标包括：开放源代码的软件发布政策、快速性能、健壮、灵活和安全。它直接同 Dan Bernstein 编写的 **qmail** 展开了竞争。所有主要的 Linux 发行版本都包含 Postfix，Mac OS X 从 10.3 版开始用它取代 **sendmail** 作为其默认的邮件系统。

对于 Postfix 来说，最重要的事情有：首先，它马上就能使用（最简单的配置文件只有一两行）；其次，它利用正则表达式映射高效地过滤电子邮件，特别是配合 PCRE（Perl Compatible Regular Expression，兼容 Perl 的正则表达式）库使用效果颇佳。Postfix 和 **sendmail** 兼容，因为 Postfix 的 **aliases** 和 **.forward** 文件与 **sendmail** 的对应文件在格式和语义上都相同。

Postfix 使用 ESMTP 协议，它也支持虚拟域和垃圾邮件过滤功能。Postfix 没有像 **sendmail** 那样使用一种重写地址的语言，相反，它依靠从纯文件、Berkeley DB、**dbm**、LDAP、NIS、NetInfo 或者 MySQL 这类的数据库进行的表查询。

18.16.1 Postfix 体系结构

Postfix 由几个小的协作程序所组成，这些程序负责发送网络消息、接收消息、在本地投递邮件等。它们之间通过 UNIX 域套接口或者 FIFO 进行通信。这种体系结构与 **sendmail** 的体系结构非常不同，**sendmail** 就在一个大程序里完成大部分工作。

master 程序启动和监视所有的 Postfix 进程。它的配置文件 **master.cf** 列出了这些辅助程序，还有应该怎样启动它们的信息。在那个文件中设定的默认值适合于除了非常慢或者非常快的机器（或者网络）之外的所有情况使用。一般而言，不需要对它进行调整。常做的一项修改是注释掉一个程序，例如 **smtpd**，在客户机不应该监听 SMTP 端口的时候这么做。

在投递电子邮件的过程中，那些最重要的服务器程序如图 18.5 所示。

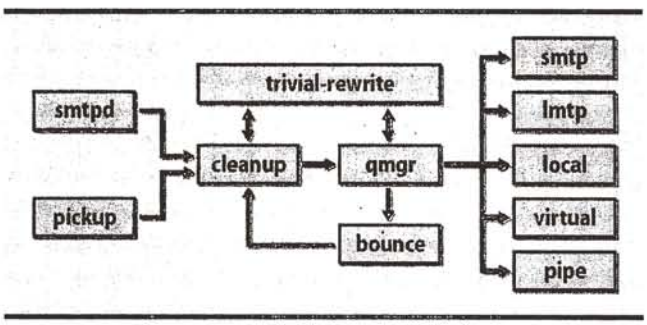


图 18.5 Postfix 的服务器程序

www.TopSage.com

18.16.2 接收邮件

smtpd 在 SMTP 端口上收邮件，它还要检查连上来的客户机是否得到授权，能够发送它们正在尝试投递的邮件。当通过**/usr/lib/sendmail** 这个 Postfix 兼容 **sendmail** 的程序往本地发邮件的时候，会在 **/var/spool/postfix/maildrop** 目录下写一个文件。**pickup** 程序定期扫描这个目录，处理它发现的任何新文件。

传入的所有电子邮件都要经过 **cleanup**，它根据 `canonical` 和 `virtual` 映射，加上缺少的信头，重写邮件地址。在把电子邮件插入到 `incoming` 队列之前，**cleanup** 要用 **trivial-rewrite** 把它略做重写，这个程序对地址做一些小的修正，例如，给不完整的地址追加一个邮件域名。

18.16.3 邮件队列管理器

等候投递的邮件由 **qmgr** 控制，这个队列管理器管理着 5 个队列：

- `incoming`（传入）——刚到的邮件；
- `active`（活动）——正在投递的邮件；
- `deferred`（推迟）——以前投递失败的邮件；
- `hold`（约束）——队列中的邮件被系统管理员阻止发送；
- `corrupt`（错误）——不可读或者不可分析的邮件。

队列管理器一般采用一种简单的 FIFO（先进先出）策略来选择要处理的下一个消息，但它也支持一种复杂的抢先式算法，该算法优先发送收件人少的消息，把有大量收件人的批量发送邮件放到后面。

为了不让接收邮件的主机忙不过来，特别是当它刚刚宕过机之后，Postfix 使用了一种慢启动算法来控制它尝试投递邮件的速度有多快。推迟发送的消息获得一个二次重发时间戳，这个时间按指数退避，从而不让无法投递的消息浪费资源。不可达的目的地会被缓存下来，以此避免做不必要的投递努力。

18.16.4 发邮件

trivial-rewrite 帮助 **qmgr** 确定应该把一则消息发到哪里。**trivial-rewrite** 做出的路由决策可以被 `transport` 映射所覆盖。

smtpd 程序负责通过 SMTP 协议把邮件投递到远程主机。**lmtp** 使用 RFC 2022 里制定的 LMTP 协议（Local Mail Transfer Protocol，本地邮件传送协议）投递邮件。LMTP 基于 SMTP，但这个协议已经做了修改，不要求邮件服务器管理邮件队列。这个邮寄程序（mailer）对于把电子邮件投递到 Cyrus IMAP 软件这样的邮箱服务器特别有用。

local 的工作是向本地投递电子邮件。它在别名表里解析地址，按照从收件人的**.forward** 文件里找到的指示处理邮件。消息不是被转发给另一个地址，传给一个外部程序处理，就是保存在该用户的邮件目录里。

virtual 这个程序把电子邮件投递到“虚拟邮箱”，也就是说，和本地的 Linux 账号没有关系的邮箱，但仍然代表有效的邮件目的地。最后，**pipe** 实现了通过外部程序投递邮件。

18.16.5 安全

Postfix 在几个层面上落实安全措施。大多数的 Postfix 服务器程序都能运行在一个 **chroot** 过的环

www.TopSage.com

境下。它们是没有父子关系的独立程序，而且它们中也没有一个是 setuid 程序。放邮箱的目录的组可写权限给了 postdrop 这个组，**postdrop** 程序的 setgid 就是这个组。

有一点令人印象深刻，除了 DoS（拒绝服务攻击）之外，任何版本的 Postfix 里尚未出现其他被利用的安全漏洞。

18.16.6 Postfix 命令和文档

用户可以通过几个命令行工具与这个邮件系统交互：

- **sendmail**、**mailq**、**newaliases**——都是和 **sendmail** 兼容的替代程序；
- **postfix**——启动和停止该邮件系统（必须以 root 身份运行）；
- **postalias**——构造、修改和查询别名表；
- **postcat**——打印队列文件的内容；
- **postconf**——显示和编辑 **main.cf** 这个主配置文件；
- **postmap**——构造、修改或者查询查找表；
- **postuper**——管理邮件队列。

Postfix 的软件发布带有一套手册页，该手册描述了所有这些程序及其选项。此外，在 www.postfix.org 的在线文档说明了配置和管理 Postfix 的各个方面。在 Postfix 的软件发布中也包含了同样的文档，所以您应该能发现它们已经装到了您的系统上，通常是在一个叫做 README_FILES 的目录下。

18.16.7 配置 Postfix

main.cf 文件是 Postfix 的首要配置文件。**master.cf** 文件配置服务器程序。它还定义了各种的查找表，供 **main.cf** 访问，并且提供不同类型的服务映射。

postconf(5)的手册页描述了能在 **main.cf** 文件中设定的每一个参数。如果您只是键入 **man postconf**，那么看到的是 **postconf** 程序的手册页。使用 **man -s 5 postconf** 才能看到描述 **main.cf** 配置选项的手册版本。

Postfix 配置语言看上去有点儿像是 Bourne shell 注释和赋值一类的语句。在变量前面加一个$，就可以在定义其他变量的时候引用它。变量定义就像它们出现在配置文件一样被保存下来，它们在使用的时候才扩展，此时要做全部的替换。

您可以通过给变量赋值的方式创建新变量。要小心选择不会和现有的配置变量冲突的名字。

包括查找表在内的所有的 Postfix 配置文件，都把以空白开头的行当作续行。这个约定能产生可读性很好的配置文件，但是您必须在第一列开始新行。

18.16.8 main.cf 的内容

在 **main.cf** 文件里可以设置超过 300 个参数。不过，在一个普通的站点上，只需要设其中的几个即可，因为默认的设定几乎都做好了。Postfix 的作者强烈建议您支配没有默认值的那些参数。那样一来，如果一个参数的默认值在将来发生了变化，那么您的配置就会自动采用新的默认值。

随 Postfix 软件发布带的 **main.cf** 文件样板包括了许多注释掉的参数例子，还有一些简要介绍。最好把原来的版本留着以供参考。从空文件开始设定您自己的配置，这样一来，您所做的设定就不会淹没在注释的汪洋大海里。

www.TopSage.com

18.16.9 基本设置

让我们从一个尽可能简单的例子开始：一个空文件。吃惊吧，这个空文件也是一个完全合理的Postfix配置文件。它配置的邮件服务器往本地投递邮件（本地的意思是指与本地主机名在同一个域），以及把目的地不是本地地址的任何消息直接发给合适的远程服务器。

另一个简单的配置是一个“空客户机（null client）”，也就是说，一个系统不向本地投递任何电子邮件，只把对外的邮件转发给一台指定的中央服务器。对于这种配置，我们定义了几个参数，首先是mydomain，它定义了主机名的域名部分，其次是myorigin，它是给不全的电子邮件地址追加的邮件域。如果mydomain和myorigin两个参数相同，那么我们就可以写成下面这个样子：

```
mydomain = cs.colorado.edu
myorigin = $mydomain
```

我们应该设定的另一个参数是mydestination，它指定了本地的邮件域（也称为“规范”域）。如果一则消息的收件人地址把mydestination作为其邮件域，那么这则消息就要通过**local**程序投递给相应的用户（假定没有找到对应的别名或者**.forward**文件）。如果在mydestination里包含了一个以上的邮件域，那么这些域都被当作是同一个域的别名。

我们想要我们的空客户机（null client）不向本地投递邮件，所以这个参数应该为空：

```
mydestination =
```

最后，参数relayhost告诉Postfix把所有不是本地的消息发给一台指定的主机，而不是把它们直接发给它们表面上的目的地：

```
relayhost = [mail.cs.colorado.edu]
```

中括号告诉Postfix把指定的字符串当作一个主机名（DNS的A记录），而不是当作一个邮件域名（DNS的MX记录）。

既然空客户机不应该接收从其他系统发来的邮件，所以在空客户机的配置里最后要做的一件事情是，注释掉**master.cf**文件里的**smptd**一行。这个改动根本不让Postfix运行**smtpd**。只用这么简单的几行配置，我们就定义了一个功能完善的空客户机。

对于一台“真实的”邮件服务器来说，您不但多需要几个配置选项，而且还要映射表。我们在接下来的几小节里介绍这些表。

18.16.10 使用 postconf

postconf是一个能帮助您配置Postfix的方便工具。不带参数运行该工具，它会打印出所有当前配置好的参数。如果您把一个特定参数作为**postconf**的命令参数，那么它就会打印出那个参数的值。**-d**选项能让**postconf**打印参数的默认值而不是当前配置的值。

例如：

```
$ postconf mydestination
mydestination =
$ postconf -d mydestination
mydestination = $myhostname, localhost.$mydomain, localhost
```

另一个有用的选项是**-n**，它让**postconf**只打印当前配置值和默认值不同的参数。如果您要在Postfix

www.TopSage.com

的邮递列表上寻求帮助，那么您应该在自己的邮件里带上这个命令输出的配置信息。

18.16.11 查找表

Postfix 操作的许多方面都是通过查找（lookup）表来决定的，这张表能把关键字映射到值，或者实现简单的列表。例如，alias_maps 表的默认设定为：

```
alias_maps = dbm:/etc/mail/aliases, nis:mail.aliases
```

数据源用 *type:aliases* 这样的表示方法指定。注意，这个特殊的表实际上同时使用了两个不同的信息源：一个是 **dbm** 数据库，另一个是 NIS 映射。多个值用逗号、空格或者两者合起来进行分隔。表 18.22 列出了能用的数据源，**postconf -m** 命令也能给出这一信息。

表 18.22 可用作 Postfix 查找表的信息源

类 型	说 明
dbm / sdbm	传统的 **dbm** 或者 **gdbm** 数据库文件
cidr	CIDR 形式的网络地址
hash / btree	Berkeley DB 散列表（代替 **dbm**）或者 B-树表
ldap	LDAP 目录服务
mysql	MySQL 数据库
nis	NIS 目录服务
pcre	兼容 Perl 的正则表达式
pgsql	PostgreSQL 数据库
proxy	通过 **proxymap** 访问，例如逃出 **chroot**
regexp	POSIX 正则表达式
static	不管关键字如何，都返回 *path* 指定的值
unix	Linux 的**/etc/passwd** 和**/etc/group** 文件，使用 NIS 句法[a]

a. unix:passwd.byname 是 **passwd** 文件，unix:group.byname 是 **group** 文件。

使用 dbm 和 gdbm 类型只是为了和传统的 sendmail 别名表保持兼容。Berkeley DB（hash）是一种更为现代的实现；它更安全，速度也更快。如果兼容性不是一个问题的话，那么使用：

```
alias_database = hash:/etc/mail/aliases
alias_maps = hash:/etc/mail/aliases
```

alias_database 指定由 **newaliases** 重建的表，它应该与您在 alias_maps 中指定的表相对应。有两个参数的原因是 alias_maps 可能包括非 DB 的信息源，例如 mysql 或者 nis，它们不需要重建。

所有 DB 类的表（dbm、sdbm、hash 和 btree）都基于一个文本文件，这个文本文件可以编译成一个搜索效率高的二进制格式。在注释和续行方面，这些文本文件的句法都和配置文件的句法相似。文件中的配置项用简单的一对“关键字/值”来指定，两者间用空格分隔，对此例外的是别名表，为了保持与 **sendmail** 的兼容性，这个表必须在关键字之后有一个分号。例如，下面这几行适合于别名表：

```
postmaster: david, tobias
webmaster:  evi
```

再举一个例子，下面是一个访问表，用于中继从任何一个主机名以 cs.colorado.edu 结尾的客户机来的邮件：

www.TopSage.com

```
.cs.colorado.edu OK
```

对于这些文本文件来说，普通表用 **postmap** 命令编译成它们的二进制格式，而别名表则用 **postalias** 命令编译成二进制格式。表的说明（包括类型）必须用第一个参数指明，例如：

```
$ postmap hash:/etc/postfix/access
```

postmap 还能查询一张查找表里的值：

```
$ postmap -q blabla hash:/etc/postfix/access
$ postmap -q .cs.colorado.edu hash:/etc/postfix/access
OK
```

18.16.12 本地投递

local 程序负责把邮件投递到合乎规定的域。它还处理本地别名。例如，如果 mydestination 设为 cs.colorado.edu，而收到一封发给 evi@cs.colorado.edu 的电子邮件，那么 **local** 首先查一次 alias_maps 表，然后递归地替换任何匹配项。

如果没有匹配的别名，那么 **local** 就找 evi 主目录里的**.forward** 文件，如果有这个文件（句法和一个别名映射的右边一样），它就按照这个文件的指示处理邮件。最后，如果没有找到**.forward** 文件，那么电子邮件就投递到 evi 的本地邮箱。

在默认情况下，**local** 把电子邮件写到**/var/mial** 下标准 mbox 格式的文件里。您可以用表 18.23 里的参数改变 **local** 的操作。

表 18.23 **local 投递到本地邮箱采用的参数（在 main.cf 里设置）**

参 数	说 明
mail_spool_directory	把邮件投递到一个服务所有用户的中央目录
home_mailbox	把邮件投递到指定的相对路径下的*~usr*
mailbox_command	用一个外部程序投递邮件，一般是 **procmail**
mailbox_transport	通过 master.cf 里定义的一个服务投递邮件[a]
recipient_delimiter	允许使用扩展的用户名（参见下面的介绍）

a. 这个选项和邮箱服务器程序（例如 Cyrus 的 **impad**）交互。

mail_spool_directory 和 home_mailbox 这两个参数正常情况下会生成 mbox 格式的邮箱，但是它们也能生成 **qmail** 风格的 **Maildir** 邮箱。在路径名末尾加一个斜线，就能按这个方式生成邮箱。

如果 recipient_delimiter 被设为+，那么可以接受发到 evi+whatever@cs.colorado.edu 这个地址的邮件，它被投递给 evi 的账号。有了这个功能，用户就能创建特殊用途的地址，并且按目的地址来对它们进行分类。Postfix 首先按完整地址来查找，只有在找不到的情况下，它才去掉扩展的部分，退而使用基本地址。Postfix 还能查找相应的转发文件**.forward+whatever** 做更进一步的别名替换。

18.16.13 虚拟域

如果您想要在 Postfix 邮件服务器上托管一个邮件域，那么您可以有 3 种选择。

- 在 mydestination 里列出这个域。邮件按照前面介绍的那样来投递：做别名扩展，把邮件投递给相应的用户。
- 在参数 virtual_alias_domains 里列出这个域。这个选项让这个域有自己的独立寻址空间，和

www.TopSage.com

系统的用户账号没有关系。在这个域中的所有地址都必须能解析成（通过映射）它之外的真实地址。

- 在参数 virtual_mailbox_domains 里列出这个域。和 virtual_alias_domains 选项一样，这个域有自己的名字空间。不过，邮件可以投递到一个指定的目录下的所有邮箱里，和用户账号没有关系。

在上述 3 个地方之一列出这个域。请仔细选择，因为许多配置元素都要依靠这个选择。我们已经看过 mydestination 方法的处理。其他几个选项在下面讨论。

18.16.14 虚拟别名域

如果一个域列为 virtual_alias_domains 参数的值，那么 Postfix 就接受发到那个域的邮件，而且必须把这些邮件转发到本地机器或者别处的一个实际的收件人那里。

对虚拟域里地址的转发功能必须在参数 virtual_alias_maps 里包括的一张查找表里定义。这张表里的表项，左边是虚拟域，右边是实际的目的地址。

右边域名不全的名字被解释为本地主机上的一个用户名。

考虑下面这个从 **main.cf** 里截取的例子：

```
myorigin = cs.colorado.edu
mydestination = cs.colorado.edu
virtual_alias_domains = admin.com
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

于是在**/etc/mail/admin.com/virtual** 里，我们会有下面几行：

```
postmaster@admin.com evi, david@admin.com
abuse@admin.com      evi
david@admin.com      david@schweikert.ch
evi@admin.com        evi
```

发给 evi@admin.com 的邮件都被重新定向到 evi@cs.colorado.edu（追加了 myorigin），最终投递到 evi 用户的邮箱，因为 mydestination 里包括 cs.colorado.edu。

定义可以递归：右边可以包括在左边再次被定义的地址。注意，右边只可以是一个地址的列表。如果您需要执行一个外部程序，或者要使用 :include: 文件，那么要把这个电子邮件重定向为一个别名，接着就可以按照您的需要对别名进行扩展。

为了把所有内容都放在一个文件里，您可以把 virtual_alias_domains 设为和 virtual_alias_maps 一样的查找表，并且在这张表里加一个特殊项，把它标为一个虚拟的别名域。在 **main.cf** 文件里：

```
virtual_alias_domains = $virtual_alias_maps
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

在**/etc/mail/admin.com/virtual** 里：

```
admin.com                    notused
postmaster@admin.com evi, david@admin.com
...
```

给邮件域（admin.com）的配置项的右边实际上从来都不用，该表中存在独立的 admin.com 一项就足以让 Postfix 把它当作一个虚拟的别名域。

18.16.15 虚拟邮箱域

在 virtual_mailbox_domains 下列出的域和本地（规范）域类似，但是用户的列表及其对应

www.TopSage.com

的邮箱必须独立于系统的用户账号进行管理。

参数 virtual_mailbox_maps 指向了一张表，其中列出了该域内所有的有效用户名。这个映射表的格式为：

```
user@domain /path/to/mailbox
```

如果这一路径以斜线结尾，那么邮箱就保存为 **Maildir** 格式。virtual_mailbox_base 的值一定以这个指定的路径开头。

您经常可能想要给 virtual_mailbox_base 中的地址起个别名。使用 virtual_alias_map 就能达到上述目的。下面是一个完整的例子。

在 **main.cf** 中的配置为：

```
virtual_mailbox_domains = admin.com
virtual_mailbox_base = /var/mail/virtual
virtual_mailbox_maps = hash:/etc/mail/admin.com/vmailboxes
virtual_alias_maps = hash:/etc/mail/admin.com/valiases
```

/etc/mail/admin.com/vmailboxes 可能包括类似下面这样的项：

```
evi@admin.com          nemeth/evi/
```

/etc/mail/admin.com/valiases 可能有下面这样的项：

```
postmaster@admin.com evi@admin.com
```

您甚至可以在不在虚拟域内的地址上使用虚拟别名映射。虚拟别名映射能让您重定向任何域的任何地址，而与这个域的类型（规范、虚拟别名或者虚拟邮箱）没有关系。因为邮箱的路径可以放在虚拟邮箱映射的右边，所以使用这个机制是在那个域中建立别名的唯一途径。

18.16.16 访问控制

邮件服务器只应该为可信的客户机中继转发邮件。如果邮件服务器把不认识的客户机发来的邮件转发给其他服务器，那么就称为开放中继（open relay），这种情况很不好。参考 18.10.1 节了解有关这个问题的详情。

还好，Postfix 默认不会做开放中继。实际上，它的默认配置相当严格，您更有可能的是要放开权限而不是紧缩权限。Postfix 中对 SMTP 处理的访问控制通过“访问限制列表”来配置。表 18.24 中的参数控制着在一次 SMTP 会话的不同阶段应该检查什么 smtpd_client_restrictions。

表 18.24 local 投递到本地邮箱采用的参数（在 main.cf 里设置）

参 数	何时应用
smpd_client_restrictions	建立连接请求的阶段
smtpd_helo_restrictions	在 HELO/EHLO 命令阶段（启动会话）
smtpd_sender_restrictions	在 MAIL FROM 命令阶段（发件人说明）
smtpd_recipient_restrictions	在 RCPT TO 命令阶段（收件人说明）
smtpd_data_restrictions	在 DATA 名阶段（邮件主体）
smtpd_etrn_restrictions	在 ETRN 命令阶段[a]

a. 这是用于重新发送队列中邮件的一个特殊命令。

最重要的参数是 smtpd_recipient_restrictions，因为当获悉收件人的地址，并且识别出

www.TopSage.com

它是否为本地地址之后最容易实施访问控制。表 18.24 中的其他所有参数在默认配置中都为空。这个默认值为：

```
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

依次测试每一项指定的限制，直到找到对该邮件的明确处理决定。表 18.25 给出了最常用的限制。

表 18.25 常见的 Postfix 访问限制

参 数	何 时 应 用
check_client_access	使用一张查找表来检查客户主机的地址
check_recipient_access	使用一张查找表来检查收件人邮件地址
permit_mynetworks	授权给 mynetworks 列出的地址
reject_unauth_destination	拒绝发给非本地收件人的邮件，不做中继

每样信息都能用这些限制来测试，不只是像 smtpd_sender_restrictions 中的发件人地址那样的特定信息。因此，为了简单起见，您可能想把所有的限制都放在一个参数之下，这个参数应该是 smtpd_sender_restrictions，因为只有这一个参数能够测试所有的信息（除了 DATA 部分）。

smtpd_recipient_restriction 是检测邮件中继的位置。您应该保留 reject_unauth_destination 这个限制，仔细选择在它之前的“允许（permit）”限制。

18.16.17 访问表

每项限制返回表 18.26 中的动作之一。在这些限制中使用访问表（例如，check_client_access 和 check_recipient_access）来分别选择一个基于客户主机地址或者收件人地址的动作。

表 18.26 访问表的动作

动 作	含 义
4nn text	返回临时错误码 *4nn* 和消息 *text*
5nn text	返回临时错误码 *5nn* 和消息 *text*
DEFER_IF_PERMIT	如果限制的结果是 PERMIT，那么把它改为一个临时错
DEFER_IF_REJECT	如果限制的结果是 REJECT，那么把它改为一个临时错
DISCARD	接受消息，但是悄悄地丢弃它
DUNNO	假装没有找到该关键字，测试更多的限制
FILTER *transport:dest*	把邮件通过过滤器 *transport:dest*[a] 传给邮件
HOLD	把邮件阻止在队列中
OK	接受这个邮件
PREPEND *header*	给消息加一个信头
REDIRECT *addr*	把这个邮件转发到一个指定的地址
REJECT	拒绝该邮件
WARN.*message*	在日志里加入给定的警告消息 *message*

a. 参考 18.16.19 节的内容。

举个例子，假定您想要给 cs.colorado.edu 域内的所有机器做中继，而且还想只让可信的客户机向内部的邮递列表 newsletter@cs.colorado.edu 发邮件。那么您应该在 **main.conf** 里用下面的几行配置实现上面两项策略：

www.TopSage.com

```
smtpd_recipient_restrictions =
        check_client_access hash:/etc/postfix/client_access
        check_recipient_access hash:/etc/postfix/recipient_access
        reject_unauth_destination
```

注意，当给一个参数指定一列值的时候，可以有逗号，也可以没有。

在**/etc/postfix/client_access** 中：

```
.cs.colorado.edu    OK
```

在**/etc/postfix/recipient_access** 中：

```
newsletter@cs.colorado.eduREJECT Internal list
```

REJECT 后的文本是个可有可无的字符串，它随同出错码一起发给客户机。它告诉发件人邮件为什么被拒绝。

18.16.18 客户机身份验证

对于从家里发邮件的用户来说，通过家里的 ISP 邮件服务器送出邮件，而不管发件人的地址是否出现在那封邮件中的做法通常最容易。大多数 ISP 信任自己的直接用户，所以允许中继。如果不可能有这样的配置，或者如果您正在使用诸如 Sender ID 或者 SPF 这样的系统，那么就要确保能授权网络之外的用户向您的 **smtpd** 提交消息。

解决这个问题的一种途径是捎带在 POP 或者 IMAP 所用的验证协议之上。需要发邮件的用户也需要读他们的邮件，所以仅一步身份验证就能实现发邮件和读邮件两方面的验证。

这种捎带系统叫做 POP-before-SMTP（先 POP 再 SMTP）或者 IMAP-before-SMTP（先 IMAP 后 SMTP），它的工作方式如下：只要用户已经由 POP 或者 SMTP 守护进程验证了身份，那么用户的 IP 地址就明确列入可以用 SMTP 发邮件的名单（whitelist，白名单）大约 30 分钟。列入这个名单的客户机的数据库由一个专门负责该任务的守护进程维护。在 Postfix 中，这样的配置类似下面：

```
smtpd_recipient_restrictions = permit_mynetworks
          check_client_access hash:/etc/postfix/pop-before-smtp
          reject_unauth_destination
```

对这个问题更好的解决方案是使用 SMTP AUTH 机制来直接在 SMTP 层面上验证身份。必须编译 Postfix 支持 SASL 库才能使用这种方法。于是您可以像下面这样配置这个功能：

```
smtpd_sasl_auth_enable = yes
smtpd_recipient_restrictions = reject_non_fqdn_recipient
        permit_mynetworks
        permit_sasl_authenticated
```

您还需要支持加密的连接，避免以明文方式发送口令。使用 2.2 版之前的 Postfix 时，要给源代码打上 Postfix/TLS 这个补丁。给 **main.cf** 文件加上下面这几行：

```
smtpd_use_tls = yes
smtpd_tls_auth_only = yes
smtpd_tls_cert_file = /etc/certs/smtp.pem
smtpd_tls_key_file = $smtpd_tls_cert_file
```

18.16.19 反垃圾邮件和病毒

Postfix 有许多功能有助于阻止可疑的电子邮件。

www.TopSage.com

有一类保护功能要求严格实现 SMTP 协议。合法的邮件服务器应该会遵守这一协议，但是垃圾邮件和病毒的发送者则往往对协议轻率行事，因而把自己暴露了出来。遗憾的是，世界上仍然有不健全的邮寄程序在处理合法的邮件，所以这项技术操作起来不一定可靠。选择限制条件时要很仔细，并且要监视日志文件。表 18.27 给出了这类功能中的一些。

表 18.27 严格检查 SMTP 协议的参数和限制

动 作	含 义
reject_non_fqdn_sender reject_non_fqdn_recipient reject_non_fqdn_hostname	拒绝发件人域名不完整、收件人域名不完整或者 HELO/EHLO 主机名不完整的消息
reject_unauth_pipelining	如果客户机没有等着看一条命令的结果状态就接着往下，则放弃当前的会话（限制）
reject_unknown_sender_domain	拒绝发件人域名不能解析的消息（限制）[a]
smtpd_helo_required	对话一开始就要求 HELO/EHLO（参数，不是 yes 就是 no）
strict_rfc821_envelopes	要求 MAIL FROM 和 RCPT TO 命令中的电子邮件地址语法正确（参数，不是 yes 就是 no）

a. 返回一个临时的出错消息，因为这个问题可能源于一次暂时的 DNS 小毛病。

在将一项限制投入正式使用之前要测试一下（始终是个不错的主意），把 `warn_if_reject` 这个限制加到它前面，把作用效果从直接拒绝转为发警告性的日志消息。

18.16.20 黑洞列表

和 **sendmail** 一样，您可以要求 Postfix 对照一个基于 DNS 的黑洞列表，检查传入的电子邮件，参考 18.10.3 节了解更多细节知识。使用限制 `reject_rbl_client`，后面跟着要咨询的 DNS 服务器的地址，就能启动这一操作。类似的，还有一个 `reject_rhsbl_sender` 限制，它检查发件人地址的域名而不是客户机的主机名。

下面的这个例子是 **main.cf** 文件中的一个相当完整的反垃圾邮件配置：

```
strict_rfc821_envelopes = yes
smtpd_helo_required = yes
smtpd_recipient_restrictions =reject_unknown_sender_domain
        reject_non_fqdn_sender
        reject_non_fqdn_recipient
        permit_mynetworks
        check_client_access hash:/etc/postfix/client_access
        reject_unauth_destination
        reject_unauth_pipelining
        reject_rbl_client relays.ordb.org
        reject_rhsbl_sender dsn.rfc-ignorant.org
```

注意，我们在 `permit_mynetworks` 之前放了一些限制。这样的调整能让我们核实我们自己的客户机是否在发送格式正确的邮件。这是一种找出配置错误的容易办法。

18.16.21 SpamAssassin 和 promail

Postfix 支持 SpamAssassin 和那一类的其他过滤器。参考 18.10.8 节和 18.9.16 节了解这些工具的一般知识。

从用户的 **.forward** 文件可以启动 **procmail**，但是这样做很复杂而且容易出错。更好的办法是在 **main.cf** 文件里加入下面一行：

www.TopSage.com

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION"
```

Postfix 接着使用 **procmail** 投递邮件，而不是直接把消息写到邮件池里去。给 **procmail** 的参数传递了地址的扩展（在+之后的部分），随后可以在 **procmail** 里用$1 访问到它。

18.16.22 策略守护进程

Postfix 的 2.1 版引入了一种将访问控制授权给外部程序的机制。这些程序称为策略守护进程（policy daemon），它们收到 Postfix 对一则邮件消息掌握的所有信息，必须返回表 18.26 列出的一种处理动作。

或许用这样的一个策略守护进程能实现的最有意思的功能就是“greylisting（灰名单技术）”。灰名单技术根据客户主机名、发件人地址和收件人地址这个三元组来对每个传入的消息进行分类。Postfix 第一次看到一个给定的三元组的时候，它返回给发件人一个临时错误消息。合法的邮件服务器会在大约 10 分钟后尝试重新投递邮件，什么时候才允许这则消息进来。因为第一次投递通常在几分钟里就做了，所以邮件不会被过分地拖延。

灰名单技术在思想上类似于 Postfix 要求严格遵守 SMTP 的功能。在使用灰名单技术的情况下，重新投递的尝试本身就体现出它是合法的邮件服务器。灰名单经证实对于清除垃圾邮件非常有效，因为许多垃圾邮件源都使用不那么高级的软件，它们不会尝试重新投递邮件。

18.16.23 内容过滤

Postfix 能够使用正则表达式检查电子邮件消息的信头和主体是否有违禁内容。它还能把消息传给其他程序，例如专门的反垃圾邮件工具或者反病毒应用。

对信头和主体的检查是随着消息通过 SMTP 接受实时执行的。做检查的每个正则表达式如果匹配，就会调用表 18.26 中指定的一个动作。例如，下面这行在 **main.cf** 文件中的配置：

```
header_checks = regexp:/etc/postfix/header_checks
```

配合**/etc/postfix/header_checkers** 里的下面这行：

```
/^Subject: reject-me/ REJECT You asked for it
```

就能拒绝标题以“reject-me”开头的任何消息。虽然支持正则表达式总是不错的，但是在处理电子邮件的情况下要澄清许多地方。具体而言，这不是一种过滤垃圾邮件或者病毒的有效方法。

功能很强的病毒过滤技术通常是通过 Amavis 实现的，Amavis 是一个 Perl 程序，它把邮件服务器软件同一个或者几个反病毒应用衔接起来。这样的过滤器用 Postfix 的 `content_filter` 参数来配置，这个参数要求 Postfix 把发来的每一则消息都穿过指定的服务。除了设定 `content_filter` 参数之外，您必须修改 **master.cf** 文件中的一些现有的配置项，并且增加一些新的配置项。Amavis 带有对此的详细指导。有许多 Amavis 的变体可用，我们推荐 Mark Martinec 编写的 **amavisd-new**。

18.16.24 调试

当您遇到 Postfix 出问题的时候，首先要检查日志文件。您问题的答案最有可能就在那儿，这只是个把答案找出来的问题。每个 Postfix 程序正常情况下都会给它所处理的每则消息生成一条日志记录。例如，一则对外发出的消息的日志记录可能看上去像下面这样：

www.TopSage.com

```
Aug 18 22:41:33 nova postfix/pickup: 0E4A93688: uid=506
    from=<dws@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/cleanup: 0E4A93688: message-id=
    <20040818204132.GA11444@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: from=<dws@ee.ethz.ch>,
    size=577,nrcpt=1 (queue active)
Aug 18 22:41:33 nova postfix/smtp: 0E4A93688:
    to=<evi@ee.ethz.ch>,relay=tardis.ee.ethz.ch[129.132.2.217],delay=0,
    status=sent (250 Ok: queued as 154D4D930B)
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: removed
```

由此可见，这则引起我们关注的信息出现了好多行。注意，标识符 0E4A93688 每行都有：只要一则消息进入邮件系统，Postfix 就给它分配一个队列 ID，并且从不改变它。因此，在日志中搜索一则消息的历史时，首先要集中精力确定该消息的队列 ID。一旦您知道了 ID，那么 **grep** 一下日志，找到所有相关的日志记录就很容易了。

Postfix 非常擅长于给它所注意到的问题记录有用的日志消息。不过，在成千上万行正常的状态消息中定位重要的消息行有时候却比较困难。这里正是使用 10.5 节介绍的日志分析工具来凝练出有用信息的好地方。

18.16.25 查看邮件队列

找问题的另一个地方是邮件队列。和 **sendmail** 系统中的情形一样，有一个 **mailq** 命令可以打印出队列中的内容。您可以用它看看一则消息是否处理不下去了，为什么会这样。

另一个有用的工具是 Postfix 最近的版本带的 **qshape** 这个脚本。它给出了有关一个队列的汇总统计数据。输出结果如下：

```
# qshape deferred
                  T  5 10 20 40 80 160 320 640 1280 1280+
      TOTAL      78  0  0  0  7  3   3   2  12    2    49
    expn.com     34  0  0  0  0  0   0   0   9    0    25
  chinabank.ph    5  0  0  0  1  1   1   2   0    0     0
 prob-helper.biz  3  0  0  0  0  0   0   0   0    0     3
```

qshape 按照收件人的域给出了指定队列的分类总结。各列报告相关消息已经在队列中等候的时间。例如，您可以看到有 49 个发往 expn.com 的消息已经在队列中等候了超过 1280 分钟的时间。这个例子中的所有目的地址都暗示出这些是用户的休假应答脚本响应垃圾邮件的消息。

qshape 也能用**-s** 标志按发件人的域来做总结。

18.16.26 soft_bounce

如果 `soft_bounce` 被设为 `yes`，那么 Postfix 对于在它正常情况下应该发送永久性错误消息（例如“user unknown”或者“relaying denied”）的时候，则发送临时的出错消息。这是一种很棒的测试功能，它能让您监视修改过配置之后的消息处理情况，而不必冒永久丢失合法邮件的风险。您拒绝的任何东西最终都会再回头试一次。但是不要忘记在您做完测试之后关闭这项功能。否则，您就不得不要一遍遍处理每一个被拒绝的消息了。

18.16.27 测试访问控制

测试访问控制限制的最简单的方法是从一台外界的主机发一个消息，看看情况如何。这是一项很

好的基本测试，但是它不能涵盖特殊的条件，例如从一个您无法登录进去的特定域发来的邮件。

Postfix 2.1 引入了一种对 SMTP 协议的扩展，称为 XCLIENT，它模拟从另一个地方提交邮件。这项功能默认是禁用的，但是在 **main.cf** 文件里采用下面的配置行，就能让从 localhost 发起的连接具有这一模拟功能：

```
smtpd_authorized_xclient_hosts = localhost
```

测试会话看上去就像下面这样：

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 tardis.ee.ethz.ch ESMTP Postfix
XCLIENT NAME=mail.cs.colorado.edu ADDR=192.168.1.1
250 Ok
HELO mail.cs.colorado.edu
250 tardis.ee.ethz.ch
MAIL FROM: <evi@colorado.edu>
250 Ok
RCPT TO: <david@colorado.edu>
554 <david@colorado.edu>: Relay access denied
```

18.17 推荐读物

COSTALES, BRYAN 和 ERIC ALLMAN. *sendmail, (3nd Edition)*. Sebastopol, CA: O'Reilly Media, 2002.

这本书是 sendmail 的权威巨著——有 1200 页之多。它既包括一部教程，又有完整的参考资料。本书适合于采用随手翻到某一页来阅读的方式，我们认为这对于一本参考书来说是至关重要的特性。它也有很好的索引。

CLAYTON, RICHARD. "Good Practice for Combating Unsolicited Bulk Email." RIPE/Demon Internet. 2000. http://www.ripe.net/ripe/docs/ripe-206.html。这份文档的内容是面向 ISP 而写的。它包含许多策略信息以及一些连到技术主题的好链接。

SCHWARTZ, ALAN. *SpamAssassin*. Sebastopol, CA: O'Reilly Media, 2005.

SCHWARTZ, ALAN 和 PAULA FERGUSON. *Managing Mailing Lists*. Sebastopol, CA: O'Reilly Media, 1998.

HAZEL, PHILIP. *The Exim Smtp Mail Server: Official Guide for Release 4*. Cambridge, UK: User Interface Technologies, Ltd., 2003.

从 www.exim.org 可以找到 Exim 的文档和资料。

sendmail 的手册页介绍了它的命令行参数。参见 Eric Allman 写的 *Sendmail: An Internetwork Mail Router* 了解梗概。

在 *Sendmail Installation and Operation Guide* 中介绍了配置文件的安装指导和描述，在 **sendmail** 软件发布的 doc/op 子目录下可以找到这份文档。它的内容相当完整，配合 **cf** 目录里的 **README** 文件一起阅读的话，它能让您对 **sendmail** 系统有详尽的了解。

www.sendmail.org、www.sendmail.org/~ca 和 www.sendmail.org/~gshapiro 都有和 **sendmail** 有关的文档、HOWTO 和教材。

RFC2822 的出现代替了作废的 RFC822。RFC2822 描述了连网的邮件系统中消息和地址的句法，

www.TopSage.com

而 RFC1123 则描述了对主机的要求。这些 RFC 在某种意义上都是构造 **sendmail** 的功能规范。

RFC2821 代替了 RFC821，它定义了 SMTP 协议（Simple Mail Transport Protocol，简单邮件传输协议），而 RFC1869、1870、1891 和 1985 把它扩展成了 ESMTP。

RFC974 描述了域名系统（Domain Name System）的 MX 记录，以及它们与邮件路由处理的关系。

- RFC1731——IMAP4 的身份验证机制；
- RFC1733——IMAP4 的分布式电子邮件模型；
- RFC2033——LMTP，本地邮件传输协议（Local Mail Transport Protocol）；
- RFC2076——常用 Internet 消息头；
- RFC2142——常用服务、角色和功能的邮箱名；
- RFC2505——SMTP MTA 反垃圾邮件的建议；
- RFC2635——不要作呕，发送大量宣传邮件的指南；
- RFC2821——简单邮件传输协议；
- RFC2822——Internet 消息格式；
- RFC4405——SMTP 服务扩展，用于指明消息的提交者；
- RFC2487——Sender ID，电子邮件身份验证；
- RFC4408——授权在电子邮件中使用域的 SPF，第 1 版；
- RFC4409——提交邮件消息。

RFC2821（SMTP）和 2822（Internet 消息格式，Internet Message Format）整理了一些最常被引用的 Internet 邮件 RFC；它们替代了 RFC821、822、974 和 1869。RFC 2821 和 2822 在 2001 年 4 月首次公布，而且被推荐作为标准。

18.18 习题

E18.1 [专门针对 **sendmail** 的问题] 简要列出 `genericstable` 和 `virtusertable` 之间的不同和相似之处。每种功能都在什么样的情况下使用。

E18.2 [专门针对 **sendmail** 的问题] 比较使用**/etc/mail/aliases** 和使用 LDAP 服务器保存邮件别名。每种做法有哪些优缺点？

E18.3 简要阐述邮件用户代理（MUA）、投递代理（DA）和访问代理（AA）的不同之处。然后说明邮件传输代理（MTA）和邮件提交代理（MSA）之间的不同。

E18.4 [专门针对 **sendmail** 的问题] **smrsh** 是什么？您为什么用它代替**/bin/sh**？如果您的站点上用到了 **smrsh**，那么哪些程序得到允许可以作为程序的邮寄代理来运行？它们中间有什么不安全的危险之处吗？

E18.5 [专门针对 **sendmail** 的问题] 写一个**/etc/mail/aliases** 小文件，演示 3 种不同类型的别名。简要叙述每行的作用以及它为什么有用。

★E18.6 简要描述一下下面这个电子邮件的信头。这封电子邮件的传送路线是什么？它的地址表明它要发送给谁？它被投递给了谁？这封电子邮件从收件人发送到目的地用了多长时间？

```
From clements@boulderlabs.com Fri Dec 28 17:06:57 2001
Return-Path: <clements@mail.boulderlabs.com>
Received: from boulder.Colorado.EDU (boulder.Colorado.EDU
    [128.138.240.1]) by ucsub.colorado.edu (8.11.6/8.11.2/ITS-5.0/student)
    with ESMTP idfBT06vF10618 for <hallcp@ucsub.Colorado.EDU>; Fri,
    28 Dec 2001 17:06:57-0700 (MST)
```

www.TopSage.com

TopSage.com

```
Received: from mail.boulderlabs.com (mail.boulderlabs.com
    [206.168.112.48]) by boulder.Colorado.EDU
    (8.10.1/8.10.1/UnixOps+Hesiod (Boulder)) with ESMTP id
    fBT06uL13184; Fri, 28 Dec 2001 17:06:56 -0700 (MST)
Received: from ath.boulderlabs.com (cpe-24-221-212-162.co.sprintbbd.net
    [24.221.212.162]) by mail.boulderlabs.com (8.11.6/8.11.6) with ESMTP
    id fBT06oQ29214 for <booklist@boulderlabs.com>; Fri, 28 Dec 2001
    17:06:50 -0700 (MST) (envelope-from
    clements@mail.boulderlabs.com)
From: David Clements <clements@boulderlabs.com>
Received: (from clements@localhost) by ath.boulderlabs.com
    (8.11.6/8.11.4) id fBT06ma01470 for booklist@boulderlabs.com; Fri,
    28 Dec 2001 17:06:48 -0700 (MST) (envelope-from clements)
Date: Fri, 28 Dec 2001 17:06:48 -0700 (MST)
Message-Id: <200112290006.fBT06ma01470@ath.boulderlabs.com>
To: boolist@boulderlabs.com
Subject: Book Questions
```

★E18.7 [专门针对 **sendmail** 的问题] 列举并说明邮件队列目录里文件前缀及其含义。为什么删除它们中间的一些很重要，删除其他一些为什么又是非常错误的？如何利用其中的一些前缀来调试 **sendmail** 的配置错误？查看一下校园网邮件服务器上的 **mailq**。在这个目录里有什么机关吗？有消息却没有控制文件的情况有吗？有控制文件没有消息的情况有吗？在队列中最老的消息是什么？（要求有 root 权限）

E18.8 看看您校园里邮件服务器上的 **mailq**。这个目录里有什么毛病吗？有没有控制文件的消息，或者没有消息的控制文件吗？队列中最老的消息是什么？（需要 root 权限）。

★E18.9 解释下面每条 **m4** 宏的用途。如果这个宏包含有一个文件，则简要说明文件的内容应该是什么。

a) `VERSIONID`

b) `OSTYPE`

c) `DOMAIN`

d) `MAILER`

e) `FEATURE`

★E18.10 说明 MX 记录是什么。为什么 MX 记录对于邮件的投递来说很重要？给出 MX 记录配置错误造成邮件无法投递的例子。

★E18.11 被列入 sbl-xbl.spamhaus.org 的黑名单，或者上了类似的一个垃圾邮件黑洞列表意味着什么？列举一些用于远离这些列表的技术。

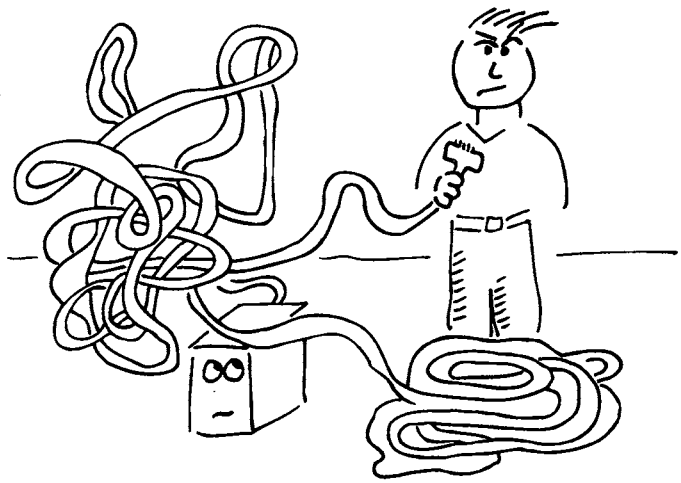
★E18.12 如果您的站点允许使用 **procmail**（哪怕我们说过不应该用它），而且您从本地的系统管理小组那里获得许可，那么设置您个人的 **procmail** 配置文件来展示 **procmail** 是怎样造成安全漏洞的。

★★E18.13 研究您的站点上目前采用的 MTA 配置。其中使用了哪些特殊的 MTA 功能？您可以找出这个配置里有什么问题吗？有办法让这个配置更理想吗？

★★E18.14 在您的邮箱里找到一封垃圾邮件，研究它的信头。报告表明这份邮件可能被伪造过的任何蛛丝马迹。然后运行本章提到的一些工具，比如 SpamCop 或者 SpamAssassin，报告它们发现的结果。您是怎样辨认出伪造出来的信头的？提交这封垃圾邮件以及您给发件人下的结论、所列主机是否有效的说明、以及看上去不对劲儿的任何别的地方。

www.TopSage.com

第 19 章　网络管理和调试



因为网络增加了机器之间的相互依赖程度，所以它们往往会使问题扩大化。正如俗话说的："连网的时候，就是仅仅因为一台机器上出现了从未听说过的故障就能使您什么也干不了的时候。"

网络管理是保持网络健康的艺术和学问。它通常包括下面这些任务：

- 网络、网关和关键性服务器的故障检测；
- 把问题通知给管理员的方案；
- 对网络进行总体监视，平衡负荷并规划扩展；
- 对网络的建档和可视化；
- 从中央站点管理网络设备。

在一个单独的网段上，一般不值得花时间建立正式的网络管理规程。只要在安装之后彻底地测试整个网络，并偶尔检查一下以确保其负载不会过重就可以了。如果它出现故障，维修该网段就是了。

随着网络的增长，管理规程应该更为自动。在由几个不同子网通过交换机或路由器连接而成的网络上，您可能想用 shell 脚本和简单的程序来启动自动化的管理工作。如果是一个 WAN 或复杂的本地网络，就应该考虑安装一台专用的网络管理工作站了。

在某些情况下，您所在单位对可靠性的需求将决定网络管理系统的复杂程度。一个网络问题可以使所有工作陷于停顿。如果您的站点不能容忍中断，那么就值得获取并安装一套高端的企业级网络管理系统。

很遗憾，即便是最好的网络管理系统也不能防止所有的故障。关键在于要有一个建立了良好文档的网络，而且有高素质的人员来处理不可避免的网络崩溃事故。

www.TopSage.com

19.1 网络故障的检测

有几种不错的工具可以用在 TCP/IP 协议层上对网络进行调试。它们大多会给出底层信息，所以为了使用这些调试工具，就必须了解 TCP/IP 和路由的主要概念。

另一方面，网络的问题也可能是由高层协议（比如 DNS、NFS 和 HTTP）的问题造成的。在阅读本章之前，可能需要您先通读一下本书的第 12 章和第 13 章。

在本节中，我们将从一些常规的故障检测策略讲起。然后我们将介绍几个基本工具，包括 **ping**、**traceroute**、**netstat**、**tcpdump** 和 **Wireshark**。在本章中我们不讨论 **arp** 命令，虽然它的确也是一个有用的调试工具——要获得更多信息请参考 12.6 节。

在对网络动手之前，请考虑如下原则。

- 每次只做一处改动，并测试每一处改动以保证它已经达到了您所希望的效果。如果没有达到预期的效果，就要撤销所做的任何更改。
- 在采取行动之前记录下当时的情况，并对整个过程中的每一处改动做记录。
- 从系统或网络的某个“端点”开始，沿着系统的关键性组件开展工作，直到搞清楚问题为止。例如，您可以从查看客户机上的网络配置开始，一直检测到物理连接，再检查网络硬件，最后还要检查服务器的物理连接和软件配置。
- 定期沟通。大多数的网络问题会牵扯或影响到许多不同的人：用户、ISP、系统管理员、电信工程师、网络管理员等。保持清晰、持续的沟通使你们不会妨碍彼此解决问题的努力。
- 作为团队开展工作。多年的经验表明，如果人们平等地相互帮助，就会少犯愚蠢的错误。
- 按网络的层次来处理问题。从“顶部”或“底部”开始，一步一步地对协议栈进行检查。

最后一点值得多讨论一下。正如 12.2 节所描述的那样，TCP/IP 的体系结构定义了几个抽象的协议层，网络的各个组成部分就在这些协议层上运行。例如，HTTP 依赖于 TCP、TCP 依赖于 IP、IP 依赖于以太网协议、而以太网协议则依赖于网络导线的完好无损。如果能首先确定出发生问题的协议层，您就可以极大地减少调试一个问题所花费的时间。

当您沿着协议栈往上（或往下）推进时，请自问一下类似下面这样的问题。

- 物理连接和链路的指示灯是否亮？
- 接口的配置正确吗？
- ARP 表能显示出其他的主机吗？
- 可以 ping 通本地主机地址（127.0.0.1）吗？
- 可以用 IP 地址 ping 通其他的本地主机吗？
- DNS 的配置正确吗？[1]
- 可以用主机名 ping 通其他的本地主机吗？
- 可以 ping 通另一个网络上的主机吗？
- 高层服务（比如 Web 和 SSH 服务器）是否能够工作？

一旦确定了问题在哪一层，就退一步想想后面的测试和修理工作会给其他服务以及主机带来什么样的影响。

19.2 ping：检查主机是否正常

ping 命令实在是太简单了，但在许多情况下，您只需要它就够了。它向目标主机发送一个 ICMP

[1] 如果您的机器在引导时挂起、引导得非常慢或者在进入 SSH 连接时挂起，应该首先怀疑 DNS。

www.TopSage.com

ECHO_REQUEST 包，并等着看那台主机是否返回应答。虽然 **ping** 很简单，但它是网络调试工作的主力之一。

您可以用 **ping** 来检查个别主机的状态和测试网段。在处理 **ping** 的过程中会涉及路由表、物理网络和网关，所以，网络必须或多或少地能够工作，**ping** 才可能成功。如果 **ping** 不通，您可以相当肯定，其他更复杂的东西就更没法运行了。不过，这条规则不适用于用防火墙封锁 ICMP 应答的网络。下结论说目标主机对 **ping** 不应答之前，请查明防火墙有没有妨碍调试。您可以考虑把爱管闲事的防火墙禁用一小段时间以帮助调试。

除非给出一个包计数参数，否则大多数版本的 **ping** 都会以无限循环的方式运行。**ping** 够了之后，键入中断字符（通常是<Control-C>）即可退出。

下面是一个例子：

```
$ ping beast
PING beast (10.1.1.46): 56 bytes of data.
64 bytes from beast (10.1.1.46): icmp_seq=0 ttl=54 time=48.3ms
64 bytes from beast (10.1.1.46): icmp_seq=1 ttl=54 time=46.4ms
64 bytes from beast (10.1.1.46): icmp_seq=2 ttl=54 time=88.7ms
^C
--- beast ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 46.490/61.202/88.731/19.481 ms
```

ping beast 的输出显示了该主机的 IP 地址、每个响应包的 ICMP 序号，以及往返所花的时间。上面的输出告诉您最明白不过的事情是：服务器 beast 是开着的，并且已连接到网络上。

在一个正常运行的网络中，**ping** 可以让您确定一台主机是否已宕机。反之，当已知一台远程主机开着并且处于良好的工作状态时，**ping** 可以为您提供有关网络健康状态的有用信息。ping 包是按照通常的 IP 机制传递的，一次成功的往返意味着源主机和目的地之间的所有网络和网关都能正确工作，至少是大致如此。

ICMP 序号是特别有价值的信息。序列的不连续说明有丢包。虽然事实上 IP 不保证包的投递，但一个健康的网络应该很少丢包。丢包的问题对于向下跟踪来说很重要，因为它们往往被更高层的协议所掩盖。网络也许看起来工作正常，但要比它应该具有的速度慢得多，这不仅是因为包的重传，还因为检测和处理它们需要有协议开销。

为了查到丢包的原因，首先运行 **traceroute**（请参见下一小节）找出发给目标主机的包的路由。然后按顺序 ping 各个中间网关，找出丢包的链路。为了查明问题，您发送包的数量必须多到有统计意义才行。网络故障通常位于最后一个能够 ping 通且包没有明显丢失的网关和下一个网关之间的链路上。

ping 报告的往返时间使您能够大致了解一条贯穿网络的路径的整体性能。往返时间中的适度偏差通常不代表有问题。包偶尔可能被没有明显理由地延迟几十或几百毫秒。这只是 IP 和 Linux 的工作方式罢了。您应该期望看到的是，大多数包的往返时间相当一致，偶然有一些偏离。现今许多路由器实现了对 ICMP 包的限速响应，这意味着如果一个路由器已经处理了许多 ICMP 流量，就可能延迟对您的 ping 的响应。

ping 程序允许您发送任何大小的回应请求包。通过使用比网络的 MTU（对以太网而言是 1500 字节）更大的包，可以强行产生分片。这种方法可以帮助您分辨出是介质错误还是其他的底层问题，比如拥挤的网络或者 VPN 所带来的问题。您还可以用**-s** 标志以字节为单位指定包的大小。

```
$ ping -s 1500 cuinfo.cornell.edu
```

使用 **ping** 命令时请牢记下面的警告。

www.TopSage.com

首先，只用 **ping** 命令很难区别出是网络故障还是服务器的故障。ping 不通只能告诉您有问题（网络防火墙有时候会有意阻断 ICMP 包）。

其次，ping 对目标机器的状态没有太多保证。回应请求包是在 IP 协议堆栈内处理的，不需要在被探测的主机上运行一个服务器进程。得到回应只能保证那台机器是开着的，并且没有处于内核恐慌中（kernel panic）。需要更高层次的方法才能验证个别服务，比如 HTTP 和 DNS 的可用性。

19.3 traceroute：跟踪 IP 包

traceroute 最初由 Van Jacobson 编写，它使您能够找出 IP 包到达其目的地所经过的一系列网关。几乎所有现代的操作系统都带有某个版本的 **traceroute**。

它的句法很简单：

```
traceroute hostname
```

这条命令有各种各样的选项，其中多数在日常使用中都不重要。和通常的情形一样，*hostname* 可以用 DNS 域名或 IP 地址指定。输出仅仅是一个主机列表，从第一个网关开始，到目的地为止。

例如，从主机 jaguar 到主机 nubark 做 **traceroute** 生成的输出如下：

```
$ traceroute nubark
traceroute to nubark (192.168.2.10), 30 hops max, 38 byte packets
 1  lab-gw (172.16.8.254)  0.840 ms  0.693 ms  0.671 ms
 2  dmz-gw (192.168.1.254)  4.642 ms  4.582 ms  4.674 ms
 3  nubark (192.168.2.10)  7.959 ms  5.949 ms  5.908 ms
```

从上面的输出中我们可以知道，jaguar 到 nubark 正好相距 3 跳，而且可以看到连接包括哪些网关。还显示了每个网关的往返时间——每跳都测量并显示 3 个样本。Internet 上两台主机之间做 **traceroute** 经常超过 15 跳。

traceroute 的工作原理是把一个发出包的存活时间（TTL，实际上是“存活的跳数”）字段人为地设置为较低的数目。在包到达一个网关时，它们的 TTL 会减 1。当某个网关把 TTL 减少到 0 时，它就丢弃这个包并把一则 ICMP “time exceeded（超时）” 消息返回给源主机。

第一次发出的 3 个 **traceroute** 包的 TTL 被设为 1。看到这样一个包的第一个网关（在本例中是 lab-gw）判断出 TTL 已超出，就送回一则 ICMP 消息，通知 jaguar 包丢失了。在发生错误的包的头部里，发送包一方的 IP 地址标识出了这个网关，**traceroute** 在 DNS 中查找这个地址来找出这个网关的主机名。参考 15.7.5 节了解有关反向 DNS 查询的更多知识。

为了识别出第二跳的网关，**traceroute** 第二轮发送的包把 TTL 字段设置为 2。第一个网关在传递包的同时把它们的 TTL 减 1。在第二个网关处，包被丢弃，生成和上次一样的 ICMP 错误消息。这个过程持续到 TTL 等于到目的地主机的跳数，从而让包成功到达它们的目的地。

大多数路由器用“最靠近”您的主机的接口发送它们的 ICMP 消息。如果从目的主机反向执行 **traceroute**，或许会看到，同一组路由器用了不同的 IP 地址来标识。您可能还会看到完全不同的路径，这样的配置称为“非对称路由”。

因为 **traceroute** 对 TTL 字段的每个值都要发送 3 个包，所以有时您可能会观察到一个有趣的人为现象。如果一个中间网关是在几条路线上多路传输流量的，包可能会被不同的主机返回，在这种情况下，**traceroute** 只是把它们都打印出来。

让我们看一个更有趣的例子，从 colorado.edu 的一台主机到 xor.com 的 **traceroute** 结果：

www.TopSage.com

大学网 TopSage.com

```
rupertsberg$ traceroute xor.com
traceroute to xor.com (192.225.33.1), 30 hops max, 38 byte packets
 1  cs-gw3-faculty.cs.colorado.edu (128.138.236.3)  1.362 ms  2.144 ms  2.76 ms
 2  cs-gw-dmz.cs.colorado.edu (128.138.243.193)  2.720 ms  4.378 ms  5.052 ms
 3  engr-cs.Colorado.EDU (128.138.80.141)  5.587 ms  2.454 ms  2.773 ms
 4  hut-engr.Colorado.EDU (128.138.80.201)  2.743 ms  5.643 ms  2.772 ms
 5  cuatm-gw.Colorado.EDU (128.138.80.2)  5.587 ms  2.784 ms  2.777 ms
 6  204.131.62.6 (204.131.62.6)  5.585 ms  3.464 ms  2.761 ms
 7  border-from-BRAN.coop.net (199.45.134.81)  5.593 ms  6.433 ms  5.521 ms
 8  core-gw-eth-2-5.coop.net (199.45.137.14)  53.806 ms  *  19.202 ms
 9  xor.com (192.225.33.1)  16.838 ms  15.972 ms  11.204 ms
```

以上输出说明包在离开网络 colorado.edu 之前必须遍历 5 个内部网关（从 cs-gw3-faculty 到 cuatm-gw）。在 BRAN 网络（204.131.62.6）上的下一跳网关没有 DNS 域名。在 coop.net 中历经两跳之后，我们到达了 xor.com。

在第 8 跳中，我们看到一个星号代替了一个往返时间。这个标记表明没有收到响应这次探测的回应（错误包）。在此情况下，原因可能是拥塞，但这不是唯一的可能性。**traceroute** 依赖的是优先级低的 ICMP 包，许多路由器都很聪明，懂得可以丢弃它而选择传输“真正的”流量。几颗星不应让您感到不安。

如果在某个给定网关的往返时间域中看到的全是星号，说明没有收到从那台机器来的“time exceeded”消息。也许这个网关仅仅是关机了。有时，网关会被配置为悄悄地丢弃 TTL 已超时的包。在这种情况下，您仍然能够穿过这台静默的主机看到前面的网关。另一个可能性是该网关的错误包返回得较慢，**traceroute** 在它们到达之前就不再等待了。

某些防火墙会完全封掉 ICMP 的“time exceeded”消息。如果在路径上有这类防火墙，您将得不到在该网关之后的任何相关信息。不过，您仍然可以确定到达目的地的总跳数，因为探测包最终将得知整个路径。同时，某些防火墙可能会封锁 **traceroute** 发出的用来触发 ICMP 回应的出站 UDP 数据报。这个问题使得 **traceroute** 根本无法报告有用的信息。

链路的速度较慢未必表示有故障。某些物理网络本身的延迟时间就长。迟滞也可能是接收包的网络有拥塞的标志，特别是如果网络使用了 CSMA/CD 技术，这种技术会反复尝试传送一个包（以太网就是一个例子）。不一致的往返时间可以为这样的假设提供佐证，因为冲突增加了网络行为的随机性。

有时，您会看到标记!N 代替了星号或往返时间。它表示当前的网关送回一则“network unreachable”（网络不可达）错误，意思是它不知道如何传递您的包。其他可能的返回结果包括：!H 代表“host unreachable”（主机不可达），!P 代表“protocol unreachable”（协议不可达）。给您任何一个这类错误消息的网关通常都是您可以到达的最后一跳。该主机通常存在路由问题（可能因为链路断开而造成这个问题）：不是因为其静态路由错误，就是因为动态协议不能传播通往目的地的可用路由。

如果 **traceroute** 看上去似乎没在为您工作（或者运行慢得令人难以置信），那就可能是因为它在试图使用 DNS 解析网关的主机名的时候超时了。如果主机上的 DNS 功能坏了，而您又从这台机器开始追踪网络，那么可使用 **traceroute -n** 要求以数字 IP 输出。这一选项将禁止使用 DNS，它可能是让 **traceroute** 在部分损坏的网络上发挥作用的唯一方法。

19.4 netstat：获得网络统计信息

netstat 提供了有关计算机连网软件状态的大量信息，包括接口的统计信息、路由信息和连接表。除了它们都和网络有关这一事实之外，实际上各组的输出并没有一个统一的主题。您可以把 **netstat** 看成是多种网络工具的一个“大杂烩”——它揭示出不适合放在别处的各种网络信息。下面我们讨论

www.TopSage.com

5种 **netstat** 最常见的用途：

- 检查接口的配置信息；
- 监视网络连接的状态；
- 标识正在监听的网络服务；
- 检查路由表；
- 查看各种网络协议运行的统计信息。

19.4.1 检查接口的配置信息

netstat -i 显示主机上每个网络接口的配置和状态。您可以把执行 **netstat -i** 当作您熟悉新机器网络配置的一种手段。再加上**-e** 选项能显示更多的细节内容。

例如：

```
$ netstat -i -e
Kernel Interface table
eth0      Link encap:Ethernet  HWaddr 00:02:B3:19:C8:82
          inet addr:192.168.2.1 Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1121527 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1138477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:7 Base address:0xef00

eth1      Link encap:Ethernet  HWaddr 00:02:B3:19:C6:86
          inet addr:192.168.1.13  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:67543 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69652 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:5 Base address:0xed00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:310572 errors:0 dropped:0 overruns:0 frame:0
          TX packets:310572 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

这台主机有两个网络接口：一个用于常规通信，第二条网络连接“eth1”用于系统管理。RX packets 和 TX packets 报告的是各个接口上从机器启动以来接收和发送的包的个数。许多不同类型的错误都被计入 **error** 一栏，显示出有个别错误是正常的。

错误应该少于相应总包数的 1%。如果出错率较高，请比较附近几台机器的出错率。某台机器上出现大量错误可能说明机器的接口或连接有问题。各处的出错率都很高，最有可能表明介质或者网络出了问题。高出错率的最常见原因之一是，自动侦测或者自动协商失败，导致以太网速度或者双工模式不匹配。

冲突表示网络的负荷过重，错误常常表示连线有问题。虽然冲突也是一种错误，但它由 **netstat** 单独统计。collisions 一栏给出了在包发送期间遇到的冲突次数[2]。用这个数字来计算导致冲突的输出包（TX packets）的百分比。在一个正常运行的网络中，冲突应该少于输出包的 3%，而超过 10%则表示有严重的拥塞问题。正确运行的全双工链路不应该出现冲突。

[2] 这个域只有在基于 CSMA/CD 的网络（比如以太网）中才有意义。

www.TopSage.com

19.4.2 监视网络连接的状态

netstat 不带参数则只显示活动的 TCP 和 UDP 端口的状态。正常情况下不显示正在等待连接（“listening”）的不活动的服务器程序，用 **netstat -a** 可以看到它们[3]。

它的输出看上去像这样：

```
$ netstat -a
Active Internet connections (servers and established)
Proto    Recv-Q  Send-Q  Local Address   ForeignAddress      State
tcp      0       0       *:ldap          *:*                 LISTEN
tcp      0       0       *:mysql         *:*                 LISTEN
tcp      0       0       *:imaps         *:*                 LISTEN
tcp      0       0       bull:ssh        dhcp-32hw:4208      ESTABLISHED
tcp      0       0       bull:imaps      nubark:54195        ESTABLISHED
tcp      0       0       bull:http       dhcp-30hw:2563      ESTABLISHED
tcp      0       0       bull:imaps      dhcp-18hw:2851      ESTABLISHED
tcp      0       0       *:http          *:*                 LISTEN
tcp      0       0       bull:37203      baikal:mysql        ESTABLISHED
tcp      0       0       *:ssh           *:*                 LISTEN...
...
```

上面的例子是在主机 otter 上运行的结果，这个结果去掉了很多内容。例如，没有显示 UDP 和 UNIX 套接口连接。上面的输出结果表明接进来一条 SSH 连接、两条 IMAP 连接、一条 HTTP 连接，接出去一条 MySQL 连接，还有一些监听其他连接的端口。

输出结果里的地址以 *hostname.service* 的形式显示，其中 *service* 是端口号。对于知名的服务来说，**netstat** 使用**/etc/services** 中定义的映射关系，用符号表示其端口。用**-n** 选项可以获得数字地址和端口号。和大多数网络调试工具一样，如果 DNS 坏了，没有**-n** 标志的话 **netstat** 用起来会很折磨人。

`Send-Q` 和 `Recv-Q` 显示在本地主机上的连接中发送和接收队列的长度。在 TCP 连接另一端上的队列长度可能并不一样。它们应该趋向于 0，至少不会一直不等于 0。当然，如果是在网络终端上运行 **netstat**，您的连接所在的发送队列可能永远不会等于 0。

连接状态只对 TCP 有意义；UDP 是一种无连接的协议。对于当前活动连接，最常见的状态是 `ESTABLISHED`（已建立），对于等待连接的服务器，最常见的状态是 `LISTEN`（不用**-a** 的话通常不显示），对于关闭过程中的连接则是 `TIME_WAIT`。

在您已经确定基础的连网设施工作正常时，上面的显示主要是用于调试更高层次的问题。它能让您验证服务器的设置是否正确，并帮助诊断某些类型的通信错误，特别是与 TCP 有关的错误。例如，一个停留在 `SYN_SENT` 状态的连接表示某个进程正在试图联络一个不存在或不可访问的网络服务进程。

如果 **netstat** 显示出了大量处于 `SYN_WAIT` 状态的连接，说明主机可能不能处理那么多数量的连接请求了。这种不足可能是由于内核调整参数的限制，或是由于恶意的洪泛攻击（flooding）。参考第 28 章了解有关内核调优的更多信息。

19.4.3 标识正在监听的网络服务

在重视安全的时代里，有一个常见的问题，即“这台机器上有哪些进程在监听网络，等待通信连接接进来？”**netstat -a** 列出了所有主动监听的端口（所有状态是 `LISTEN` 的 TCP 端口，还可能是所

[3] UNIX 域套接口的连接也会显示出来，但是由于它们和连网没有关系，所以我们在这里不讨论它们。

www.TopSage.com

有 UDP 端口），但是在一台繁忙的机器上，由于在 **netstat** 的结果中会显示大量已经建立的 TCP 连接，所以这些显示行可能会淹没在其中。使用 **netstat -l** 可以只查看正在监听的端口，其输出格式与 **netstat -a** 相同。

您可以加上**-p** 标志，让 **netstat** 标识出和每个监听端口相关联的特定进程。下面的例子显示出了 3 种常见的服务（**sshd**、**sendmail** 和 **named**），之后还有一个不常用的服务：

```
$ netstat -lp
...
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      23858/sshd
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN      10342/sendmail
udp        0      0 0.0.0.0:53              0.0.0.0:*                           30016/named
udp        0      0 0.0.0.0:962             0.0.0.0:*                           38221/mudd
...
```

这里的 PID 为 38221 的 mudd 正在监听 UDP 端口 962。根据您的站点对用户安装软件所做的规定，您可以进一步追查这个服务。

19.4.4 检查路由表

netstat -r 将显示内核的路由表。下面的例子来自一台带有两个网络接口的 Red Hat 机器（输出的样子随 Linux 发行版本的不同而略有不同）：

```
$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.2.5.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0         40 eth0
...
```

目的地和网关既可以显示为主机名，也可以显示为 IP 地址，**-n** 标志则要求输出数字 IP。

`Flags` 表示路由的特征：`U` 表示 up（活动的），`G` 表示网关，而 `H` 表示主机路由。`U`、`G` 和 `H` 在一起，表示经过中间网关的主机路由。`D` 标志（没有显示出来）表示由 ICMP 重定向产生的路由。其余的域给出了路由上的统计信息：当前使用此路由的 TCP 连接的数目、发送包的数目、以及所使用的接口。请记住，在不同操作系统中，这个输出稍有差别。参考 12.5.1 节了解有关路由表的更多知识。

使用这种形式的 **netstat** 来检查机器的路由表是否正常。验证系统有一条默认路由并且是正确的，这一点尤其重要。全为 0 的目的地址（0.0.0.0）代表默认路由。也有可能没有一条默认路由项，但是这样的配置非常少见。

19.4.5 查看各种网络协议运行的统计信息

netstat -s 将输出散布在网络代码各个角落里的计数器的内容。输出中 IP、ICMP、TCP 和 UDP 分成单独的小节。下面是来自一台网关机器的 **netstat -s** 的输出片段，已经把它们编辑为只显示信息中最吸引人的片段。

```
Ip:
    671349985 total packets received
    0 forwarded
    345 incoming packets discarded
```

www.TopSage.com

```
    667912993 incoming packets delivered
    589623972 requests sent out
    60 dropped because of missing route
    203 fragments dropped after timeout
```

要检查保证没有包丢失或者丢弃。可以允许传入的包丢几个，但是在丢包这个指标上数值增长很快的话，则往往表明内存不足或者有其他资源的问题。

```
Icmp:
    242023 ICMP messages received
    912 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 72120
        timeout in transit: 573
        echo requests: 17135
        echo replies: 152195
    66049 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 48914
        echo replies: 17135
```

在这个例子中，在 input 部分中的回应请求（echo requests）数应该和 output 部分中的回复回应（echo replies）数一致。请注意，即使所有的包表面上看都是可以转发出去的，仍然有可能产生“destination unreachable”消息。坏包最后会到达一个拒绝它们的网关，然后错误消息将沿着网关链送回。

```
Tcp:
    4442780 active connections openings
    1023086 passive connection openings
    50399 failed connection attempts
    0 connection resets received
    44 connections established
    666674854 segments received
    585111784 segments send out
    107368 segments retransmited
    86 bad segments received.
    3047240 resets sent
Udp:
    4395827 packets received
    31586 packets to unknown port received.
    0 packet receive errors
    4289260 packets sent
```

最好是能够对这些统计信息的正常范围建立起一种感觉，这样您就可以分辨出不正常的状态了。

19.5 SAR：检查工作接口的活动

发现网络问题的一种好方法是检查眼下正在发生什么事情。过去 5 分钟里在一个接口上发出了多少包？多少字节？有冲突或者错误发生吗？上述所有问题要通过观察工作接口的活动才能回答。

在传统的 UNIX 系统上，人们选择 **netstat -i** 作为完成这一任务的工具。遗憾的是，在 Linux 上，**netstat** 报告工作接口活动的能力被破坏了。我们建议采用一种完全不同的工具：**sar**（我们将在 25.3.6 节以普通系统监视的角度讨论 **sar**）。大多数 Linux 发行版本默认都没有安装 **sar**，但是它肯定是一种

www.TopSage.com

可选的软件包。

使用 **sar -n DEV 2 30** 这条命令，就可以让 sar 每 2 秒报告一次接口活动状况，连续报告一分钟时间（也就是说报告 30 次）。参数 **DEV** 本身是一个关键字，不是设备或者接口名的占位符。

输出结果包括按字节数和包数统计的当前和平均的网络接口利用率。下面的例子选自一台有两个物理网卡的 Red Hat 机器。第二个物理网卡（eth1）显然没在用。

```
17:50:43  IFACE rxpck/s  txpck/s  rxbyt/s  txbyt/s  rxcmp/s  txcmp/s  rxmcst/s
17:50:45     lo    3.61     3.61   263.40   263.40     0.00     0.00     0.00
17:50:45   eth0   18.56    11.86  1364.43  1494.33     0.00     0.00     0.52
17:50:45   eth1    0.00     0.00     0.00     0.00     0.00     0.00     0.00
```

前两列给出了数据采样的时间和网络接口的名字。接下来的两列分别为接口收发的包数。`rxbyt/s` 和 `txbyt/s` 两列可能最为有用，因为它们给出了实际使用的带宽。最后 3 列是压缩包的统计（`rxcmp/s`、`txcmp/s`）和多播包的统计（`rxmcst/s`）。

sar -n DEV 对于追究出错来源特别有帮助。**ifconfig** 能向您报告有问题，但它不能告诉您错误是因为一个持续的底层问题引起的，还是由于一次短暂但严重的事件导致的。在各种网络负载条件下长期观察网络，能够加深您对网络运行情况的认识。在观察 **sar -n DEV** 输出的同时，可以 **ping** 大包。

19.6 包探测器

tcpdump 和 Wireshark 属于同一类工具，即通常所说的包探测器。它们监听网络上的流量，并记录下或打印出符合用户所指定的某些标准的包。例如，可以检查某台特定主机发送或接收的所有包，也可以检查与一条特定网络连接相关的 TCP 包。

包探测器可以用于解决已知问题和发现全新的问题。偶尔“嗅”一下网络以确保通信的状态良好，这是个好主意。

因为包探测器需要能够把本地机器通常不接收（或者至少是不注意）的流量都拦截下来，所以下层的网络硬件必须能接触到每一个包。从这一点来说，像以太网这样的广播技术就可以，而某些类型的令牌环网络也没问题。

因为包探测器需要尽可能多地看到原始的网络流量，所以它们可能会受到网络交换机的阻挠，按照设计，交换机会试图限制“不必要的”包的传播。不过，在交换网络上试着用探测器仍然可能获得丰富的资料。您可能发现与广播或者多播包有关的问题。根据交换机供应商的不同，您可能会惊讶于可以看到如此之多的流量。有关网络交换机的更多信息请参见 14.2.4 节。

除了有能力接触到所有的网络包之外，接口硬件还必须提供一种方法，把那些包实际传输到软件层。在正常情况下，硬件要检查包的地址，只有广播/多播包以及地址是本地主机的包才会被转发给内核。在“混杂模式（promiscuous mode）”下的接口能让内核读取网络上的所有包，甚至包括发给其他主机的包。

包探测器理解标准网络服务使用的许多种包的格式，它们通常可以用人们读得懂的形式把包打印出来。这种能力使得跟踪两个程序之间的会话流变得更加容易。某些探测器除了包头之外还打印包中的 ASCII 内容，这对探查高层协议很有用。因为一些协议用纯文本方式透过网络发送信息（甚至是口令），所以您必须学会小心从事，以避免侵犯您的用户的隐私。

我们举例的各个 Linux 发行版本都带有一个包探测器。包探测器必须从一个原始（raw）类型的网络设备读取数据，所以它必须以 root 权限来运行。虽然以 root 权限运行的局限性降低了普通用户监听网络流量的机会，但这的确不是多大的障碍。有些网点选择从大多数主机上删除探测器程序，降低它

www.TopSage.com

们被滥用的机会。如果没有别的什么要注意的了，那么应该检查系统上的接口，确保它们都没有在您不知道或者不同意的情况下以混杂模式运行。在 Linux 系统上，处于混杂模式接口会在它的 **ifconfig** 状态输出中显示 PROMISC 标志。您还可以用 `PromiScan`（可以从 www.securityfriday.com 得到）这样的工具程序检查网络上正运行在混杂模式下的接口。

19.6.1 tcpdump：探测器之王

tcpdump 是 Van Jacobson 编写的另一个惊人的网络工具，它随大多数 Linux 的发行版本一起发布。它成为业界标准的探测器程序已经有很长时间了。大多数别的网络分析工具都能读写“tcpdump 格式”的记录（trace）文件。

在默认情况下，**tcpdump** 监听它遇见的第一个网络接口。如果它选择了错误的接口，您可以用**-i** 标志强行指定接口。如果 DNS 不能用，或者只是不希望 **tcpdump** 进行名字查找，请使用**-n** 选项。这个选项很重要，因为较慢的 DNS 服务会导致过滤器在 **tcpdump** 仍可以处理包之前就开始丢包。**-v** 标志增加您看到的有关包的信息，**-vv** 则提供更为详细的数据。最后，使用**-w** 标志，**tcpdump** 可以把包存入一个文件，而用**-r** 标志可以再把它们读回来。

例如，下面的输出来自机器 nubark。过滤器的规则 **host bull** 限制只显示与机器 bull 直接相关的包，bull 不是作为源地址就是作为目的地。

```
# sudo tcpdump host bull
12:35:23.519339 bull.41537 > nubark.domain: A? atrust.com. (28) (DF)
12:35:23.519961 nubark.domain > bull.41537: A 66.77.122.161 (112) (DF)
```

第 1 个包显示 bull 向 nubark 发送了一个有关 atrust.com 的 DNS 查找请求。回应是那个 atrust.com 所对应的 IP 地址，即 66.77.122.161。注意左边的时间戳，**tcpdump** 能够理解应用层协议（本例中是 DNS）。bull 的端口是任意的，现在是数字 41537，但是因为服务器的端口号（53）是知名端口，所以 **tcpdump** 显示了它所代表的名字（“domain”）。

包探测器能够生成大量信息——量大到不但人没法跟得上，就连下层的操作系统都支持不住。为了避免在繁忙的网络上发生这样的问题，**tcpdump** 能让您指定相当复杂的过滤器。例如，下面的过滤器只采集从一个特定子网来的 Web 流量：

```
# sudo tcpdump src net 192.168.1.0/24 and dst port 80
```

tcpdump 的手册页包含了几个有高级过滤功能的例子，还有一份完整的原语指令清单[4]。

19.6.2 Wireshark：可视化的探测器

如果您更倾向于使用一种点鼠标的程序来探测包，那么 Wireshark 可能就是您需要的。从 www.wireshark.com 可以得到 Wireshark，它采用 GNU 的 GPL（General Public License，通用公共许可证）来发布，是一种基于 GTK+（GIMP 工具包）的 GUI 包探测器，它比大多数商业探测软件产品的功能还要多。您可以在自己的 Linux 机器上运行 Wireshark，或者如果您的膝上机还痛苦地陷入 Windows 的黑暗时代里，那么您也可以下载用于 Windows 的二进制版本。

除了探测包之外，Wireshark 还有两种功能，让它格外好用。一种很好的功能就是，Wireshark 可以读写大量其他格式的包记录文件，包括（但不只限于）：

[4] 如果您的过滤功能超出了 **tcpdump** 的能力，那么可以考虑使用 **ngrep**，它可以按照包的内容对包进行过滤。

www.TopSage.com

- TCPDUMP;
- NAI 的 Sniffer;
- Sniffer Pro;
- NetXray;
- Snoop;
- Shomiti Surveyor;
- 微软的 Network Monitor;
- Novell 的 LANalyzer;
- Cisco Secure IDS iplog。

第二种特别方便的功能就是，您可以点击一条 TCP 流（stream）中的一个包，让 Wireshark“重组”（接合起来）这个 TCP 流中所有包的净荷数据。如果您想要迅速搞清在一次完整的 TCP 对话中（比如在网络上承载电子邮件消息的一条连接）传送的数据是什么，这项功能就能派上用场了[5]。

Wireshark 带有抓包过滤器，它的功能和 **tcpdump** 一样。但要注意，使用 Wireshark 的一个重要难点在于，增加了“显示过滤器”这一功能，这会影响到您看到的结果，而不是探测器实际抓到的包。奇怪的是，显示过滤器使用的语法和抓包过滤器截然不同。

Wireshark 是一种功能异常强大的分析工具，几乎每个网络专家的工具包里都会包括它。此外，对于那些刚开始研究分组网络技术的人来说，Wireshark 也是一种帮助学习的好工具。Wireshark 的帮助菜单提供了许多入门的好例子。不要害怕做实验！

19.7 网络管理协议

网络的规模和价值在最近的 10 年都有了飞速的增长，伴随着网络增长而来的是需要一种高效的方法来管理它们。商业公司和标准化组织已经以许多不同的方式来解决这个难题。最重要的进展是引入了几个标准的设备管理协议和大量使用那些协议的高水准产品。

网络管理协议提供了探测一台设备以发现其配置、健康状况和网络连接的标准方法。另外，它们允许修改这些信息中的一部分，以便可以在不同类型的设备上实现标准化的网络管理，而且能在一个中心位置上执行网络管理。

最常用于 TCP/IP 的管理协议是 SNMP（Simple Network Management Protocol，简单网络管理协议）。名虽如此，实际上 SNMP 却相当复杂。它定义了一个用于管理数据的层次结构的名字空间，以及一种在每个节点上读取和写入数据的方法。它还定义了一种被管实体（“代理[agent]”）把事件通知消息（“陷阱[trap]”）发送给管理工作站的方法。

SNMP 协议本身很简单，SNMP 的大部分复杂性都在协议层之上，也就是构造名字空间的规定，以及围绕 SNMP 产生的过分而不必要的词汇（比如 protective shell）。只要别把 SNMP 的内部机制想得太难，那么 SNMP 用起来就很简单。

另外还有其他几个悬而未决的标准。它们中的许多来源于 DMTF（Distributed Management Task Force，分布式管理任务组），该任务组负责 WBEM（Web-Based Enterprise Management，基于 Web 的企业管理）、DMI（Desktop Management Interface，桌面管理接口）和 CIM（Conceptual Interface Model，概念接口模型）等概念。这些概念中的一些（特别是 DMI）已经被几家主要的厂商采纳，并可能成为 SNMP 的有益补充（甚至是替代）。不过眼下绝大多数的网络和 Linux 系统管理工作是通过 SNMP 进行的。

[5] 您可以使用 **tcpflow** 这个工具程序在命令行对 **tcpdump** 的记录文件执行类似的操作。

www.TopSage.com

因为 SNMP 只是一个抽象的协议，所以您还需要一个服务器程序（“代理[agent]”）和一个客户机（“管理器[manager]”）才能使用它（也许与直觉相反，SNMP 的服务器端代表被管理的事物，而客户机端则是管理者）。客户机从简单的命令行实用程序到专用的管理工作站都有，后者用图形的方式来显示网络，用刺眼的颜色显示故障所在。

专用的网络管理工作站是管理协议存在的主要理由。大多数产品能让您建立网络的拓扑模型和逻辑模型，这两个模型都在屏幕上呈现，并连续指示出各网络组件的状态。

正如图表可以揭示一页数字背后隐藏的含义那样，网络管理工作站可以把大型网络的状态总结为一种人脑易于接受的形式。这种管理性的总结几乎不可能以任何其他方法获得。

用协议进行管理的主要优点在于，它将各种网络硬件推向了同一个运行环境。Linux 系统基本上都是相似的，而路由器、交换机和其他底层网络组件就不是这样了。有了 SNMP，它们都说同一种语言，可以从中央位置进行探测、复位和配置。所有的网络硬件有统一的接口，这一点很好。

19.8 SNMP：简单网络管理协议

当 20 世纪 90 年代早期 SNMP 开始广泛使用时，它引发了一次小小的淘金热。数以百计的公司发布了 SNMP 管理软件包。另外，许多硬件和软件供应商都把 SNMP 代理作为它们产品的一部分。

在我们深入研究 SNMP 详细内容之前，应该注意到，与它相关的术语是在网络技术领域中最混乱的。SNMP 概念和对象的标准名称会使您不知所云。对事情的这种状态负有责任的人一定是让他们的键盘喝醉了。

19.8.1 SNMP 的组织结构

SNMP 数据以一种标准化的层次结构进行布置。这种强制的组织方式使数据空间既保持了通用性又保持了可扩展性，至少在理论上如此。这种结构的大部分都留给将来扩充，而特定厂商添加的部分则被本地化以避免冲突。命名的层次结构由 MIB（Management Information Bases，管理信息库）组成，它是描述通过 SNMP 可访问的数据的结构化文本文件。MIB 包含了对特定数据变量的说明，数据变量用被称为对象标识符（OID）的名字来引用。

用语言来表达，这意味着 SNMP 定义了一个变量的层次型名字空间，变量的值与系统“令人感兴趣的”参数相关。一个 OID 只是给一个特定被管信息命名的别致方法。

SNMP 层次结构非常像一个文件系统。不过，它是用点（.）作为分隔符，每个节点被赋予了一个数字而不是名字。根据约定，为了易于引用，还可以给节点赋予文字名称，但这种命名其实只是为高层应用方便而已，而不是层次结构的一项功能（它在原理上类似于主机名到 IP 地址的映射）。

例如，指示系统正常运行时间的 OID 是 1.3.6.1.2.1.1.3。这个 OID 也有一个人们可以读的名称：

iso.org.dod.internet.mgmt.mib-2.system.sysUpTime

SNMP 层次结构的顶层是人为的政策性数据，通常不包含有用的数据。实际上，目前只能在 OID iso.org.dod.internet.mgmt（用数字表示是 1.3.6.1.2）之下找到有用的数据。

用于 TCP/IP 的基本 SNMP MIB（MIB-I）定义了对公共管理数据的访问：有关系统、它的接口、地址转换和协议操作（IP、ICMP、TCP、UDP 及其他）的信息。稍后对这个 MIB 更完整的修订（名为 MIB-II）在 RFC1213 中有定义。大多数提供 SNMP 服务器的供应商都支持 MIB-II。表 19.1 提供来自 MIB-II 名字空间的节点样本。

www.TopSage.com

表 19.1 从 MIB-II 中选择的 OID

OID[a]	类 型	内 容
system.sysDescr	字符串	系统信息：厂家、型号、操作系统类型等
system.sysLocation	字符串	机器的物理位置
system.sysContact	字符串	机器拥有者的联系信息
system.sysName	字符串	系统名称，通常是 DNS 全名
interfaces.ifNumber	整数	出现的网络接口数量
interfaces.ifTable	表	有关每个接口的 infobits 表
ip.ipForwarding	整数	如果系统是网关则为 1，否则为 2
ip.ipAddrTable	表	IP 寻址数据表（掩码等）
ip.ipRouteTable	表	系统的路由表
icmp.icmpInRedirects	整数	接收到的 ICMP 重定向的数量
icmp.icmpInEchos	整数	接收到的 ping 的数量
tcp.tcpConnTable	表	当前 TCP 连接的表
udp.udpTable	表	有服务器监听的 UDP 套接口的表

a. 相对于 iso.org.dod.internet.mgmt.mib-2。

除了基本 MIB 之外，还有用于各种硬件接口和协议的 MIB。有用于个别厂商的 MIB 和用于特殊硬件产品的 MIB。还有您的 MIB，我的 MIB，到处都能找到 MIB。

MIB 只是一个给管理数据命名的约定。SNMP 名字空间和设备实际状态之间的映射关系必须由代理端代码支持才有用。标准 Linux 代理程序都带有用于基本 MIB（现在是 MIB-II）的代码。有些代理程序有扩展性，可以包含补充的 MIB，有些代理程序则不能。

19.8.2 SNMP 协议操作

基本的 SNMP 操作只有 4 种：get（获取）、get-next（获取下一个）、set（设置）和 trap（陷阱）。

get 和 set 是从节点读取数据和把数据写入节点的基本操作，节点用特定的 OID 标识。get-next 用于在 MIB 层次结构上步进，它也能读取表格的内容。

trap 是从服务器（代理程序）到客户机（管理器）的一个主动提供的异步通知，它报告出现了感兴趣的事件或状态。SNMP 已经定义了若干标准的 trap 操作。包括“I have just come up”（我刚启动，开始工作）通知、报告网络链路有故障或已修复的 trap、以及用于各种路由和身份验证问题的 trap。其他许多不那么标准的 trap 也很常用，包括一些只是监视其他 SNMP 变量的值并在超过指定范围时发出一则消息的 trap。指定 trap 消息目的地的机制取决于代理程序的实现。

因为 SNMP 消息有可能会修改配置信息，所以需要一些安全性机制。最简单版本的 SNMP 安全性是基于 SNMP“群体名（community name）”的概念的，它其实只是“口令”的一种极其晦涩的说法。通常有一个群名用于只读访问，另一个群名则允许写入。

尽管许多单位仍然在使用最早的基于群体名的身份验证方法，但 SNMP 标准的第 3 版引入了安全性更高的访问控制方法。虽然配置这一安全性更好的版本要多花一点儿力气，但是对于降低的风险来说还是值得的。如果因为某种原因不能使用 SNMP 第 3 版的安全机制，至少也要选择一个难猜的群体字符串。

19.8.3 RMON：远程监视 MIB

RMON MIB 允许收集一般性的网络性能数据（也就是说，不与任何特定设备相关的数据）。可以在网络中到处部署网络探测器或“探针”，以搜集有关网络使用情况和性能的信息。搜集到许多有用的数据之后，关于这些数据的统计信息和令人感兴趣的信息就可以送回中央管理工作站用于分析和显示。

www.TopSage.com

大家网 TopSage.com

许多探针有一个包捕获缓冲并可以提供一种远程 **tcpdump** 功能。

RMON 在 RFC1757 中定义，它在 1995 年成为一个标准草案。MIB 被分为 9 个“RMON 组”。每个组包含一套不同的网络统计信息。如果您的网络很大并且有许多 WAN 连接，则应该考虑购买探针来减少 WAN 链路上的 SNMP 通信量。一旦您有权使用来自 RMON 探针的统计汇总，通常就不需要从远程搜集原始数据了。许多交换机和路由器支持 RMON，至少会保存一些网络统计信息。

19.9 NET-SNMP 代理程序

在 SNMP 首次成为标准的时候，卡内基梅隆大学（Carnegie Mellon University）和 MIT 都开发出了它的实现。CMU 的实现更完整，很快就成了事实上的标准。可是当 CMU 活跃的开发工作渐渐平息下来以后，UC Davis（加州大学戴维斯分校）的研究人员就接管了这个软件。在代码稳定之后，他们把以后的维护工作放到了 SourceForge 库上。这个包现在就是 NET-SNMP。

NET-SNMP 软件发布现在对于 Linux 来说是经过授权的免费 SNMP 工具。它包含有一个 SNMP 代理程序、一些命令行工具和一个库，这个库可用于开发支持 SNMP 的应用软件。我们在这里详细讨论代理程序，而在 19.10.1 节我们会介绍命令行工具。它的最新版本可以从 Web 站点 net-snmp.sourceforge.net 得到。

和其他实现里的代理程序一样，NET-SNMP 的代理程序也负责收集有关本地主机的信息，并且通过网络把这些信息提供给 SNMP 管理器。默认的安装包括用于网络接口、内存、磁盘、进程和 CPU 统计的 MIB。代理程序很容易扩展，因为它能执行任何 Linux 命令，并把命令的输出作为 SNMP 响应返回。您可以使用这一功能来用 SNMP 监视系统上的几乎任何东西。

在默认情况下，安装的代理程序是**/usr/sbin/snmpd**。它通常在系统引导时启动，并且从**/etc/snmp** 目录下的文件里读取它的配置信息。这些文件中最重要的要算 **snmpd.conf** 文件，它包含了大多数的配置信息，而且还启动了一系列收集样本数据的方法。虽然 NET-SNMP 开发者的意图似乎只是让用户编辑 **snmpd.local.conf** 文件，但是您必须至少编辑一次 **snmpd.conf** 文件，把任何您不打算用到的默认数据收集方法禁用掉。

NET-SNMP 的 **configure** 脚本指定了一个默认的日志文件，还有两种别的本地设置。您可以使用 **snmpd -l** 来指定一个日志文件来代替默认的文件，或者指定**-s** 来把日志消息定向到 syslog。表 19.2 给出了 **snmpd** 最重要的标志清单。我们建议您一定要用**-a** 标志。为了进行调试，您应该使用**-V**、**-d** 或者**-D** 标志，它们中的每一种都能提供更多的信息。

表 19.2 NET-SNMP snmpd 的有用标志

标　志	功　能
-l *logfile*	把信息记入日志文件 *logfile*
-a	把所有 SNMP 连接的地址记入日志
-d	把每个 SNMP 包的内容记入日志
-V	启动详细的日志功能
-D	把调试信息记入日志（许多调试信息）
-h	显示 snmpd 的所有参数
-H	显示所有的配置文件指令
-A	追加到日志文件而不是覆盖它
-s	把日志记入 syslog（使用 daemon 设备）

值得一提的是，有许多和 SNMP 有关的 Perl 模块，它们都很有用。如果您有兴趣编写自己的网络

www.TopSage.com

管理脚本，可以参考 CPAN[6]了解最新的信息。

19.10 网络管理应用程序

我们在这一节里首先研究最简单的 SNMP 管理工具：和 NET-SNMP 软件包一起提供的命令。这些命令对于帮助您熟悉 SNMP 很有用，它们还能对特定的 OID 进行很出色的一次性检查。接下来，我们来看看 Cacti 这个工具，它能生成美观的 SNMP 历史数据图，还有 Nagios，一个基于事件的监视系统。我们将就购买商业系统时应该注意什么提出一些建议来结束本节的内容。

19.10.1 NET-SNMP 工具

即使您的系统有自己的 SNMP 服务器，您可能仍然想要从 NET-SNMP 的软件包编译和安装表 19.3 里列出的客户机端工具。

表 19.3 NET-SNMP 软件包中的命令行工具

命 令	功 能
snmpdelta	一直监视 SNMP 变量中的变化
snmpdf	通过 SNMP 监视远程主机的磁盘空间
snmpget	从一个代理得到一个 SNMP 变量的值
snmpgetnext	从序列中获得下一个变量
snmpset	设置代理上的一个 SNMP 变量
snmptable	得到一个 SNMP 变量表
snmptranslate	搜索并说明 MIB 结构中的 OID
snmptrap	生成一次 trap 告警
snmpwalk	从某个特定的 OID 开始遍历 MIB

这些程序除了在命令行上使用很有价值之外，它们在简单的脚本中使用也极其方便。经常用到的是让 **snmpget** 每隔几分钟把感兴趣的数据值保存在一个文本文件中(使用 **cron** 来实现这种时间安排，参考第 8 章)。

snmpwalk 是另一种有用的工具。这条命令从一个指定的 OID 开始（或者按默认情况从 MIB 的开头处开始）重复向一个代理发出“get next”调用。这样做会得到所有可用 OID 及其关联值的完整清单。当您想要从企业管理工具中找出要监视的新 OID 时，使用 **snmpwalk** 特别方便。

下面是主机 tuva 上运行 **snmpwalk** 的结果片段。“secret813community”是群体字符串，而**-v1** 指定了简单的身份验证功能。

```
$ snmpwalk -c secret813community -v1 tuva
SNMPv2-MIB::sysDescr.0 = STRING: Linux tuva.atrust.com 2.6.9-11.ELsmp #1
SNMPv2-MIB::sysUpTime.0 = Timeticks: (1442) 0:00:14.42
SNMPv2-MIB::sysName.0 = STRING: tuva.atrust.com
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:11:43:d9:1e:f5
```

[6] CPAN 代表 Comprehensive Perl Archive Network（综合性 Perl 存档网络），它收集了大量有用的 Perl 模块，非常不错。它的地址是 www.cpan.org。

www.TopSage.com

```
IF-MIB::ifPhysAddress.3 = STRING: 0:11:43:d9:1e:f6
IF-MIB::ifInOctets.1 = Counter32: 2605613514
IF-MIB::ifInOctets.2 = Counter32: 1543105654
IF-MIB::ifInOctets.3 = Counter32: 46312345
IF-MIB::ifInUcastPkts.1 = Counter32: 389536156
IF-MIB::ifInUcastPkts.2 = Counter32: 892959265
IF-MIB::ifInUcastPkts.3 = Counter32: 7712325
...
```

在这个例子里，我们看到了有关系统的一些一般性的信息，后面跟着主机的网络接口 lo0、eth0 和 eth1 的统计信息。根据您正在管理的代理所支持的 MIB 不同，完整的输出可以有数百行之多。

19.10.2 SNMP 数据的采集和绘图

与网络相关的数据最好以包含历史信息的图形方式展示出来。重要之处在于要有某种办法连续记录性能指标，并绘制成图，选择什么样的软件来做这件事则并不关键。

MRTG 是以前最流行的 SNMP 轮询和绘图软件包之一，它由苏黎士的 ETH（译者注：即著名的苏黎士理工）Tobi Oetiker 编写。MRTG 大部分是用 Perl 编写的，由 **cron** 定期执行，能够从任何 SNMP 数据源采集数据。这个程序每次运行的时候，都会保存新数据并创建新图。

最近 Tobi 把他的精力集中到了 RRDTool 上，这是一种用于保存性能指标数据并绘制成图的应用工具包。包括我们最喜欢的 Cacti 在内的所有开放源代码监视方案都基于 RRDTool。

Cacti（从 www.cacti.net 可以获得这个软件）提供了几项吸引人的功能。首先，它实现了一个无需维护、大小固定的数据库，这个软件保存的数据正好够画图。例如，Cacti 可以在一天内每分钟保存一个样本，一周内每小时保存一个样本，一年内每星期保存一个样本。这样的归并方案既能让您保存重要的历史信息，同时又不必保存不重要的详细数据，或者在管理数据库上浪费时间。

其次，除了许多别的性能指标之外，Cacti 还能记录任何 SNMP 变量的数据，并绘制成图。您可以随意采集您想要的数据。在同 NET-SNMP 代理配合使用的情况下，Cacti 可以为几乎任何系统或网络资源提供历史全貌。

图 19.1 举了 Cacti 所画图的几个例子。这些图显示出一台服务器在几周时间里的平均负载，以及一个网络接口上一天的流量情况。

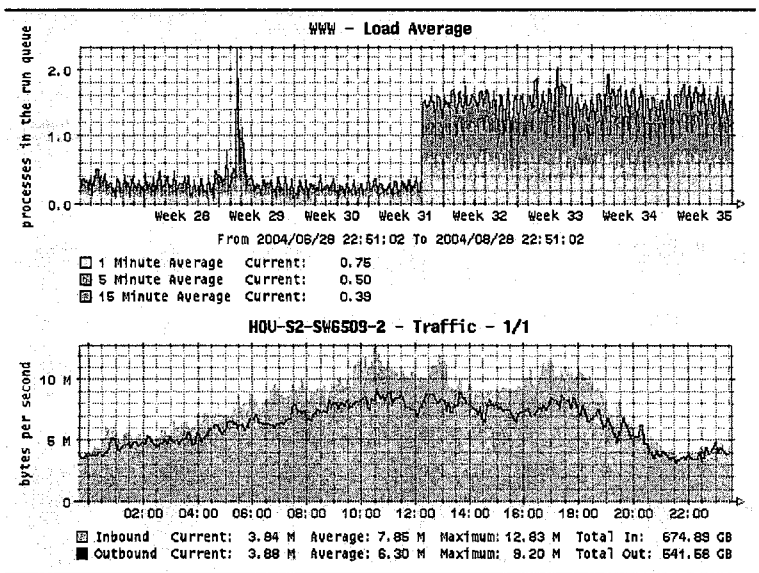


图 19.1 Cacti 画图的例子

www.TopSage.com

Cacti 基于 Web 的配置界面很好用，而且它还具有 RRDTool 自身的其他所有好处，比如维护工作量小、画图效果美观等。参见 Tobi Oetiker 的 RRDTool 主页 www.rrdtool.org，获得 RRDTool 和 Cacti 当前版本以及数十种其他监视工具的下载链接。

19.10.3 Nagios：基于事件的 SNMP 和服务监视工具

Nagios 专门用于实时故障报告。它能轮询任何 SNMP 指标（以及数百种其他网络设备）并就规定好的出错状态向您发出警报。虽然 Nagios 不能帮助您判断上个月的网络带宽增长了多少，但是它能在 Web 服务器宕机的时候呼叫您。

Nagios 软件包括很多插件，它们能监视各种各样的常见故障点。如果您觉得有必要，那么还能用 Perl 甚至 C 编写新的监视器。就通知方法而言，这个软件能发电子邮件、产生 Web 报告、使用拨号调制解调器呼叫您。和监视插件一样，用您自己的插件也很方便。

除了发送实时的服务中断报警通知之外，Nagios 还会保存历史数据。它提供了几种功能很强的报告界面，能够跟踪可用性和性能的变化趋势。许多机构使用 Nagios 测定网络是否达到 SLA 的要求，图 19.2 显示了一台 DNS 服务器的可用性情况。

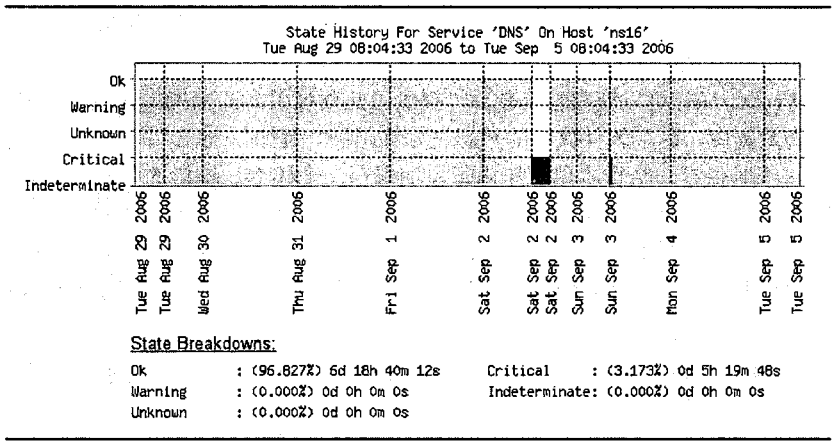


图 19.2 Nagios 显示的服务器可用性

对于包含上千台主机和设备的网络来说，Nagios 都能胜任。定制和扩展 Nagios 也很容易，它包含诸如冗余性、远程监视和渐近提醒等强大功能。如果您买不起商业性质的网络管理工具，那么应该多考虑一下 Nagios。您可以从 www.nagios.org 了解到更多内容。

19.10.4 商业管理平台

成百上千的公司在销售网络管理软件，而且每周都有新的竞争者进入市场。我们不想推荐当前最热门的产品（它在这本书印刷之前可能就不复存在了），而是试图把您在网络管理系统中要找寻的功能提出来。

数据收集的灵活性：对于管理工具而言，重要的是能够从 SNMP 之外的来源收集数据。许多软件包都包括了从所有网络服务搜集数据的能力。例如，一些软件包可以进行 SQL 数据库查询、检查 DNS 记录、以及连接到 Web 服务器。

用户界面品质：昂贵的系统通常会提供一个自定义的 GUI 或者一个 Web 接口。现在卖得最好的软件包都在鼓吹其发掘了 XML 模板用于数据呈现的能力。虽然 UI 常常似乎更像是市场营销的骗局，

www.TopSage.com

但是拥有一个可以清晰、简单和全面地呈现信息的接口是非常重要的。

价格：一些管理软件包的价格令人咋舌。HP 的 OpenView 既是最昂贵的，也是使用最广泛的网络管理系统之一。对于许多公司而言，能够说自己的站点是用高端商业系统管理的，这本身就是一种价值。如果这对贵单位不重要，您可以看看另一个极端，像 Cacti 和 Nagios 这样的免费工具。

自动发现：许多系统提供了“发现”网络的能力。通过把广播 ping、SNMP 请求、ARP 表查找和 DNS 查询组合起来，它们能够识别出所有的本地主机和设备。我们见过的所有发现实现效果都很不错，但没有一个在复杂的（或者有厚厚防火墙的）网络中能做到非常精确。

报告功能：许多产品可以发送报警邮件、激活呼机，并自动为流行的故障跟踪系统生成故障工单。要确保您选择的平台考虑到了灵活的报表功能，谁知道几年后您将要对付什么样的电子设备呢？

配置管理：一些供应商的步子远远超出了监视和告警。它们能够管理实际的主机和设备的配置。例如，CiscoWorks 提供的一个接口，能让您除了用 SNMP 监视其状态之外，还能够修改路由器的配置。因为设备配置信息涉及对网络问题更深入的分析，所以我们预言许多软件包将来会沿着这些方向发展下去。

19.11 推荐读物

CISCO ONLINE. *SNMP Overview. 1996.* www.cisco.com/warp/public/535/3.html.

MAURO, DOUGLAS R. 和 KEVIN J. SCHMIDT. *Essential SNMP (2nd Edition).* Sebastopol, CA: O'Reilly Media, 2005.

SIMPLEWEB. *SNMP and Internet Management Site.* www.simpleweb.org.

STALLINGS, WILLIAM. *Snmp, Snmpv2, Snmpv3, and Rmon 1 and 2, Third Edition.* Reading, MA: Addison-Wesley. 1999.

您会发现下面的 RFC 也很有用处。我们没有援引 RFC 的实际标题，而是描述它们的内容。实际的标题是由时髦话和 SNMP 专门术语组织起来的混合体，缺乏指导意义。

- RFC1155 SNMP 数据空间的特性（数据类型等）；
- RFC1156 MIB-I 的定义（实际 OID 的说明）；
- RFC1157 简单网络管理协议；
- RFC1213 MIB-II 的定义（OID）；
- RFC3414 SNMPv3 基于用户的安全模型；
- RFC3415 SNMPv3 基于视图的访问控制模型；
- RFC3512 用 SNMP 配置设备（最好的概述）；
- RFC3584 不同版本的 SNMP 实际共存。

19.12 习题

E19.1 在诊断一个网络问题的时候，**netstat -rn** 提供给您下面这样的输出。问题在哪儿？要用什么命令来解决它？

```
Destination     Gateway      Genmask         Flags MSS Window irtt Iface
128.138.202.0   0.0.0.0      255.255.255.0   U      40 0          0 eth0
127.0.0.0       0.0.0.0      255.0.0.0       U      40 0          0 lo
```

www.TopSage.com

★E19.2 编写一个脚本，监视一组给定的机器，如果某台机器有一段时间没有响应 ping 了就通过电子邮件通知系统管理员。不要在代码中列出主机、通知用的电子邮件地址或者用来判断没有响应的时间量。

★E19.3 做个试验，改变本地网络上某台机器的网络掩码。它仍然能工作吗？可以访问到网点上的各个地方吗？其他机器能访问到它吗？发广播能工作吗（例如，ARP 请求或者 DHCP 发现包）？解释您的发现（要求 root 权限）。

★E19.4 用 **traceroute** 命令找出您的网络上的路由通路。

a）离开您的网络要多少跳？

b）您有账号的两台机器之间有路由器吗？

c）您能发现任何瓶颈吗？

d）您的站点有多条网络连接吗？

★★E19.5 设计一个 MIB，包含您作为 Linux 系统管理员想要查询或者设置的所有变量。给 MIB 留下多种扩展途径，以包含您所忘记的重要的系统管理变量。

★★E19.6 用 **tcpdump** 命令捕获能够演示下列协议的流量。对于 TCP 会话来说，要包含并指出发起和结束包。提交一份整洁而且有良好格式的 **tcpdump** 输出（要求有 root 权限）。

a）ARP；

b）ICMP 回应请求（echo request）和响应；

c）SMTP；

d）FTP 和 FTP-DATA；

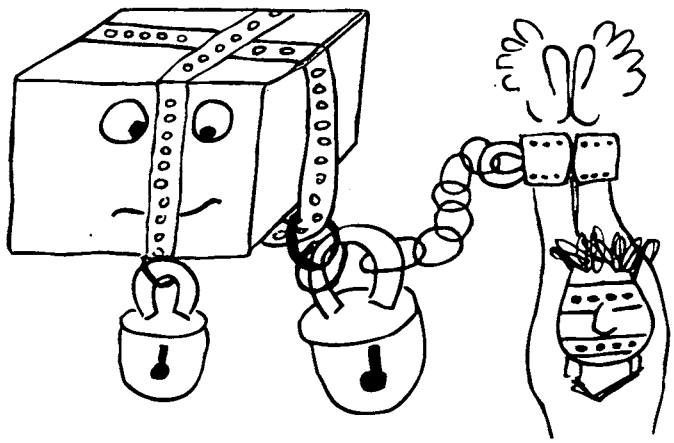
e）DNS（称为 domain）；

f）NFS。

★★E19.7 配置一幅 Cacti 图，显示在一台本地路由器上进出传输的包。这个题目要求有一个 SNMP 软件包来查询路由器，您必须知道路由器的只读群体字符串（community string）。

www.TopSage.com

第 20 章　安全



1983 年由马修·布鲁德里克（Matthew Broderick）主演的电影《战争游戏（WarGames）》为我们展现了早期计算机黑客亚文化的迷人一面。在这部电影里，大卫·莱特曼（David Lightman）从使用几种著名的攻击技术，包括战争拨号、社交工程、电话盗用（也叫做 phreaking）和口令猜测开始，最后阻止了一场核战争。《战争游戏》让计算机安全得到了公众的瞩目，而让“黑客”这个叫错的名字成为制造麻烦的青少年的代名词。这部早先的电影还只是这个当今全球性问题的序幕而已。

5 年以后，当 Robert Morris, Jr. 向全人类展示出他的“Internet 蠕虫（Internet Worm）”病毒的时候，计算机和网络技术的世界就进入了一个新纪元。在此之前，Internet 都生活在一个“纯真年代”。那时的系统管理员大多是以“如果有了再说”这样的态度来对待安全这个话题的。重大的安全事件通常由类似于这样的事情所构成：某个用户得到了系统管理员的访问权限，偷偷读了另一个用户的邮件，而这往往只是为了证明他能办得到。

Morris 蠕虫只是给系统管理员造成了成千上万小时的时间损失，但它大大地增强了 Internet 上的安全意识。我们又再次痛苦地想起了“只有好篱笆才有好邻居”这句话。结果，出现了一批供系统管理员使用的优秀工具，还有一个处理这类安全事件的正式机构。

CSI/FBI（译者注：Computer Security Institute/Federal Bureau of Investigation，美国联邦调查局/计算机安全研究所）2006 年的“计算机犯罪和安全调查报告（*Computer Crime and Security Survey*）”[1]公布的结果显示，大部分单位只把不到 5%

[1] 这是一项每年都做的调查，在 www.gocsi.com 可以看到这份调查。

www.TopSage.com

的 IT 预算用于安全。结果，因计算机安全泄露造成的损失达到了惊人的 54 494 290 美元（每家单位 174 103 美元）。这些数字反映出单位里安全意识的薄弱让人震惊。当然，花钱也不一定就能取得相应的系统安全，一位警惕的系统管理员就能在其份内的工作中阻止任何攻击分子。

总的说来，安全不是您从某个第三方商家那里买一台设备或者买一项服务就能得到的东西。商业的产品和服务只是网点安全解决方案的一部分，但它们不是包治百病的万灵药。

要让网点的安全水平达到可以接受的程度，需要有大量的耐心、警惕性、知识和持之以恒的态度——不仅您和其他系统管理员是这样，所有的用户和管理层的人员都要这样。作为系统管理员，您个人必须担负起保障系统安全，正确培训用户的责任来。您应该熟悉当前的安全技术，积极关注有关安全的邮递列表，而在问题超出了您的知识范围之后，要雇佣专业的安全专家来提供帮助。

20.1 Linux 安全吗

不！Linux 并不安全。在网络上进行通信的其他任何操作系统也都不安全。如果您确实要有绝对完全、不会被入侵的安全性，那么需要让您的计算机和其他任何设备隔开一段空间[2]。有人说，您还需要把您的计算机关在一间特殊的屏蔽了电磁辐射的房间里（用 Google 搜索一下“Faraday cage[法拉第笼]”）。好玩儿吗？

采取一些措施就能够让您的系统更能抵御攻击。即便如此，Linux 模型中几个根本性的缺陷还是不会让您达到安全无忧的桃花源。

- 和 UNIX 一样，Linux 也为使用上的方便做了优化，但是却没有让安全变得容易和自然。Linux 的思想着重考虑的是，在一个连网的多用户环境下很容易地操控数据。
- 除非经过仔细地实现，否则 Linux 的安全事实上是二元的：您要么是一个没有权限的普通用户，要么就是 root（超级用户）。幸亏有了美国国家安全局的 SELinux（Security-Enhanced Linux，增强安全的 Linux）这个项目，才在实现更细粒度的访问控制上向前取得了一些进展。但是在大多数情况下，在安全上轻微的失误就可以使整个系统遭受威胁。
- Linux 发行版本是由一大群程序员共同开发的。他们在经验水平、对细节的关注程度和在 Linux 系统及其相互依赖性方面的知识上的水平参差不齐。因此，哪怕是最善意的新功能都可能带来很大的安全漏洞。

另一方面，既然 Linux 的源代码谁都可以得到，成千上万的人都可以细细品味每一行源代码，从中找出可能会威胁安全的地方。人们普遍认为，比起封闭的操作系统来说，这种状况会显著提高安全性，因为有机会仔细研究后者的代码并找出漏洞的人在数量上非常有限。

许多站点用的还是一两个发布版之前的版本，原因不是本地化问题太麻烦，就是他们没有定购厂商的软件维护计划。无论哪种情况，即便有了针对安全漏洞的补救办法，给黑客留下的入侵机会仍然不会一夜之间就消失。

随着时间的推移，Linux 的安全性似乎应该逐渐得到改善，因为安全问题会不断被发现和纠正，可惜事实似乎并非如此。系统软件正在变得愈发复杂，黑客也变得越来越有组织性，计算机在 Internet 上的连接却变得越来越密切。安全是一场正在来临的战役，而且我们决不可能真正打赢这场战役。

还要记住：

$$安全=\frac{1}{(1.072)\times(方便)}$$

[2] 当然，无线连网技术的出现带来了一整套新问题。文中提出的隔开一段空间，意思是“干脆不连网”。

www.TopSage.com

您的系统越安全，您和您的用户就会觉得越不舒服。在仔细考虑过对您的用户造成的潜移默化的影响之后，再去实现本章所建议的安全措施。

20.2 安全性是如何受损害的

本章讨论一些常见的 Linux 安全问题和它们的标准对策。但在我们开始详细介绍之前，应该更全面地看一下真实世界中的安全性问题往往会如何发生。大多数的安全疏漏都源自于以下的某一类问题。

20.2.1 社交工程

计算机系统的用户（和系统管理员）经常是安全链条中最薄弱的一个环节。即使在有着高度安全意识的今天，毫无戒备的好心用户很容易受骗而泄露敏感信息。

这个问题呈现出许多种表现形式。攻击者冷不防给受害人打电话，装作是迷惑不解的合法用户，企图获得访问系统的帮助。系统管理员在排除故障的时候不小心把敏感信息公布到了公开论坛上。看上去没问题的维护人员在重新给电话交换间布线的时候亲手实施入侵。

“phishing（钓鱼[仿冒]）”这个术语指通过欺骗邮件、及时消息或者甚至是手机的 SMS 消息，从用户那里搜集信息的行为。仿冒特别难防，因为交流中经常包含受害人的特定信息，从而让攻击者的身份很像是真的。

社交工程一直是一种强大的黑客技术，也是最难以抵御的威胁之一。您站点的安全政策应该包括对新雇员的培训。为了提供有关电话禁忌、人身安全、邮件假冒和口令选择方面的信息，在全单位内定期进行交流是一种有效的方法。

要衡量贵单位对社交工程的抵御能力，您会发现自己做一些社交工程的攻击就能获得结果。但要保证在这么做之前，先从自己的上级那里明确获得允许。如果没有得到明确授权，这样的攻击企图将显得非常令人怀疑。它们也是内部的间谍行为的一种形式，所以如果它们没有光明正大地去实施的话，就有可能引起愤恨情绪。

20.2.2 软件漏洞

多年以来，在计算机软件（包括来自第三方的软件，商业的和免费的软件）中已经发现了不计其数能够削弱安全性的缺陷（bug）。黑客利用编程中的细微错误或者上下文依赖关系，已经能够控制 Linux，让它做任何他们想让它做的事情。

缓冲区溢出是一种常见的编程错误，也是一种牵扯到复杂因素的错误。开发人员经常预先分配一定量的临时内存空间，称为一个缓冲区，用以保存特殊信息。如果代码没有仔细地把要存放的数据大小同应该保存它的空间大小进行对照检查，那么靠近该分配空间的内存就有被覆盖的风险。熟练的黑客输入仔细组织过的数据就能导致程序崩溃，更糟糕的话能执行代码。

幸亏近年来缓冲区溢出攻击发生的显著数量提高了编程界对这个问题的认识。虽然缓冲区溢出仍然在出现，但它们经常会被很快发现和纠正，特别在开放源代码应用中更是如此。像 Java 和.Net 这样的比较新的编程体系都包含自动检查数据大小，防止发生缓冲区溢出的机制。

缓冲区溢出是称为输入有效性漏洞的更大一类软件安全缺陷下的一个子类。几乎所有的程序都接受某种类型的用户输入（例如，命令行参数或者 HTML 表格）。如果代码没有严格检查输入的格式和内容是否合适，就开始处理数据，那么就会发生问题。考虑下面这个简单的例子：

www.TopSage.com

```
#!/usr/bin/perl
# Example user input validation error

open (HTMLFILE, "/var/www/html/$ARGV[0]") or die "trying\n";
while (<HTMLFILE>) { print; }
close HTMLFILE;
```

这段代码的意图可能是打印**/var/www/html** 下某个 HTML 文件的内容，**/var/www/html** 是 Red Hat 服务器上 Apache 的 **httpd** 服务程序默认的文档根目录。这段代码接受用户输入的一个文件名，把它作为参数传给 open 函数。不过，如果一个恶意的用户把**../../../etc/shadow** 当作参数输入的话，就会显示出**/etc/shadow** 的内容！

作为一名系统管理员，您该如何防止这种情况发生呢？几乎无可奈何，至少在确定出缺陷（bug），并发布解决它的补丁之前是这样。紧盯补丁和安全性公告的发布，这是大多数系统管理员工作的一个重要部分。大多数 Linux 发行版本都带自动打补丁的工具，如 Fedora 上的 **yum**、Debian 和 Ubuntu 上的 **apt-get**。利用这些工具能让您的站点避免出现软件的安全漏洞问题。

20.2.3 配置错误

许多软件既能配置得安全，也能配置得不那么安全。遗憾的是，因为开发软件是为了方便使用，所以默认情况下经常是配置得不安全。黑客经常会利用软件的一些特性来获得访问权，在没有那么多奸诈的平和环境下，这些软件功能被认为是有帮助和方便的。没有口令的账号、与全世界共享的磁盘、以及没有保护措施的数据库，都是这种情况的一些例子。

主机配置上常出现一个安全漏洞，让 Linux 系统不要引导加载程序的口令就能引导，这是一种标准的做法。LILO 和 GRUB 在安装时都能配成有口令，但是系统管理员几乎总是忽略这个选择。这种忽视让系统向物理攻击敞开了大门。不过，它也是一个体现要在安全和方便之间取得平衡的例子。要口令意味着如果系统无意间重启了（例如，掉电之后），系统管理员就不得不亲自到现场让机器恢复运行。

保障系统安全最重要的步骤之一仅仅是确保您没有无意间给黑客留下可乘之机。这类问题虽然可能很多，而且也不一定明显知道要检查什么，但却最容易发现并改正。本章后面将介绍端口和漏洞扫描工具，它们可以帮助一位积极主动的系统管理员在问题被利用之前先发现和改正它们。

20.3 认证和标准

如果本章的标题有点儿让您感到退缩，别担心。计算机安全是一个复杂和广阔的话题，无数书籍、Web 网站和杂志都是这么说它的。幸好人们已经做了大量工作来帮助把获得的资料加以量化和组织。现在有几十种标准和认证，尽职尽责的 Linux 系统管理员应该认识到它们的指导作用。

在信息安全领域中最基本的思想原则之一被不正式地称为“CIA 安全三原则（CIA triad）”。

这个首字母缩写词代表：

- 机密性（Confidentiality）；
- 完整性（Integrity）；
- 可用性（Availability）。

数据机密性关注数据的保密。实质上，应该把对信息的访问限制在获得授权的人手里。身份验证、访问控制和加密是机密性的几个组成部分。如果一个黑客侵入了 Linux 系统，窃取了包含客户联络信息的数据库，就意味着发生了破坏机密性的事件。

www.TopSage.com

大家网 TopSage.com

完整性同信息的真实性有关系。数据完整性技术要保证信息合法有效，且不会以未经授权的方式被修改。它也规定了信息来源的可信度。但一个安全的 Web 网站提供了一个签发的 SSL 证书，它就向用户证明了，不仅它发送的信息是加密的，而且有一个可信的 CA（certificate authority，认证中心），比如 VeriSign 或者 Equifax 已经核实了信息源的身份。PGP 和 Kerberos 这样的技术也能保障数据的完整性。

信息必须在授权用户需要的时候才提供给他们，否则持有信息就没什么作用。不是由入侵者造成的中断，比如由管理上的错误或者电力中断造成的中断，都归入可用性问题的一类。遗憾的是，在问题发生之前，可用性总是被忽视。

只要您设计、实现或者维护 Linux 系统，就要考虑 CIA 安全三原则。正如安全方面的老话说的那样，“安全是一个过程”。

20.3.1 认证

这个 CIA 速成仅仅是对更大的信息安全领域的一个简要介绍。大企业经常雇用许多全职员工，他们的工作就是保卫信息安全。为了获得专业领域内的资格认证和掌握最新知识，这些职员会参加培训课程和取得证书。就像我们取得了最流行的几种证书那样，您可以自己准备。

认可范围最广的安全证书之一是 CISSP，即信息系统安全认证专家（Certified Information Systems Security Professional）。该认证由 $(ISC)^2$（International Information Systems Security Certification Consortium，国际信息系统安全认证协会）主持。CISSP 的诱人之处之一就是 $(ISC)^2$ 的“共识”（CBK，common body of knowledge）概念，CBK 实质上是信息安全领域内得到业界广泛使用的最佳做法。CBK 涵盖法律、密码、身份验证、物理安全等。对于安全人员来说，它是一个参考宝库。

一直以来，对 CISSP 就存在这样的批评，即它太注重覆盖面的广度，因而缺乏深度。CBK 里有那么多的专题，而时间又那么短！为了解决这个问题，$(ISC)^2$ 推出了 CISSP 在体系结构、工程和管理领域的专项（concentration）认证。这些专门性的证书给 CISSP 通用证书增加了深度。

SANS（System Administration，Networking，and Security，系统管理、网络和安全）协会在 1999 年创立了 GIAC（Global Information Assurance Certification，全球信息保障证书）系列认证。30 多项独立的考察覆盖了整个信息安全领域，考试分为 5 类。证书按难度分，有中级的 GISF（两次网上考试）、专家级的 GSE（23 小时的网上考试）。GSE 有业界最难认证考试之一的美誉。其中的许多考试都着重技术规范，只要求一点点经验。

最后，CISA（Certified Information Systems Auditor，信息系统认证审计师）资格是一个审计和处理证书。它的重点在业务连续性、规程、监视和其他管理内容。有些人认为 CISA 是一个中级证书，适用于单位里负责安全工作的领导角色。它最吸引人的方面之一是它只包括一项考试。

虽然这些证书都是发给个人的，但毫无疑问，它们都能应用到公司业务上。现在，越来越多的公司把证书当作专家的标志。许多公司都给予有证书的雇员更多的关注和更高的职位。如果您决定拿一个证书，可以同您的单位密切结合，让单位帮助支付相关的费用。

20.3.2 标准

因为对数据系统的依赖性与日俱增，所以已经颁布了法律和规章制度来规范对敏感、重大信息的管理工作。美国主要的立法项目，如 HIPAA、FISMA 和 SOX（Sarbanes-Oxley Act，萨班斯-奥克斯利法案）都已经包含了有关 IT 安全的内容。虽然这些法律提出的要求有时候落实起来很花钱，但它们确实让这个一度被忽视的技术方面得到了应有的重视。

遗憾的是，这些规定充满了法律术语，难以解释。大多数都没有具体说明如何达到要求。结果，

www.TopSage.com

人们又制定了很多标准，帮助系统管理员达到极高的法律要求。这些标准并不针对法律规定，但是遵守标准的话，就能保证符合法律。一下子面对所有这些不同标准的要求有点儿吓人，但这些计划起到了一种指导作用。

ISO/IEC 17799 标准可能是世界上接受面最广的标准。它首先在 1995 年作为一种英国标准出现，标准长达 34 页，分为 11 个部分，涉及从政策到物理安全再到访问控制的各个领域。每个部分的目标规定了特定的要求，在每个目标下的控制方法说明了建议的“最佳实践”方案。

这些要求不涉及技术，任何单位都能以符合自身需要的方式实现要求。但缺点在于，标准里不具体的文字说明让读者有一种灵活性很强的感觉。批评人士指出，不具体的规定让实施标准的单位向攻击敞开了大门。

尽管如此，这个标准仍然是信息安全界最有价值的文件之一。它在管理和工程之间架起一座切实的桥梁，帮助双方将力量集中在把单位风险降至最低上面。

PCI DSS（Payment Card Industry Data Security Standard，支付卡业数据安全标准）是一种完全不同的标准。在银行卡交易爆炸性的发展之后，它唤起了业内需要加强安全的意识。例如，在 2005 年 6 月，国际信用卡系统服务公司（Card-Systems Services International）披露“丢失”了 4 千万个信用卡号。

据美国国土安全部估计，仅 2004 年一年，因身份盗窃而造成的损失就高达 526 亿美元。当然，这些损失不都和信用卡被盗直接有关，但是商家不断增加的警惕性肯定起到了正面作用。FBI 甚至把信用卡诈骗联系到了恐怖分子的资金来源上。具体的事件包括在巴厘岛和马德里地铁发生的炸弹爆炸案。

尽管 PCI DSS 标准现在由 Visa 维护，但它是 Visa 和 Mastercard 之间共同努力的结果。和 ISO 17799 不同，任何人都能免费下载获得它。它完全集中在保护持卡人数据系统上，该标准有 12 个部分，规定了保护要求。

因为 PCI DSS 的重点在信用卡处理机构，所以它不适合于不涉及信用卡数据的商业领域。不过，对于那些涉及信用卡数据的领域来说，为避免严厉的罚款和可能的犯罪指控，必须严格遵守标准。您可以在 Visa 网站上的商品安全部分找到这份文件。

许多站点都编制了自定义的软件，用来满足自己独一无二的需求，但这往往会以安全为代价。容易出现缓冲区溢出、SQL 注入和跨站点脚本化攻击，这些都是自编软件常见缺点的几个例子。还好，美国政府颁布了一个规范文件，帮助检验这类应用软件。国防部的“Application Security Checklist（应用软件安全复查表）”是一份全面的指南，它能指导如何检测应用软件，发现其安全弱点。

和本节讨论的其他标准不同，“Application Security Checklist（应用软件安全复查表）”深入具体的技术，有时候还触及为执行某项测试而应该运行的各条命令。该标准的可读性没有这里提到的其他标准那么好，也没有那么正式，但是它对于需要内部开发的站点特别有用。

20.4 安全的技巧和思想

本章讨论的安全问题范围很广，种类很多。理想情况下，您应该在您的环境里解决所有这些问题。不过，如果您的时间或者耐性不够的话，这里有 6 条综合起来看是要给予考虑的最重要的安全问题，外加一些附带的要遵守的规则（大多数系统管理员都应该切实消化本章的全部内容，或许还要不只一次地这样做）。

20.4.1 包过滤

如果您把 Linux 系统通过 Internet 接入连接到网络上，您必须在 Linux 系统和外界之间有一台过滤

www.TopSage.com

数据包的路由器或者防火墙。另一种替代的做法是，使用 Linux 系统自带的 **iptables** 配置包过滤机制（在 20.13 节讨论）。无论采用什么样的实现，包过滤器都应该只允许这样的流量通过：它们的目的地是您专门想在这台 Linux 系统上提供的必要服务。

20.4.2 不必要的服务

在默认安装中打开什么网络服务这个问题上，各种版本的 Linux 的认识有很大的不同。大多数发行版本都带有几种不必要的服务。因此，要由您来考察系统上已经启用的各种服务，并且关闭没有绝对必要的那些服务。从 **netstat** 和 **fuser** 这两条命令开始是一条不错的途径。为了切实搞清楚一个攻击者能看到哪些服务，可以从一台远程主机用端口扫描程序对您的系统进行扫描。20.10.1 节介绍的 **nmap** 就是一个安全工程师的理想端口扫描程序。

20.4.3 软件补丁

所有的主流 Linux 发行版本都会定期发布与安全有关的软件包，通常每月数次。您必须紧密地关注与您所用的系统软件（以及您正在运行的任何软件）相关的补丁，并且立即安装它们。要记住，在补丁出现之前，“坏小子们”已经知道这个漏洞好几个星期了。

20.4.4 备份

不要太指望 RAID 盘阵，而是要把数据中心的数据备份到别处去。您必须定期对您所有的系统做备份，这样一来，如果发生了安全事件，您就能够有效地恢复系统。镜像、RAID 或者“热待机（hot standby）”技术都不足以取消备份。第 9 章里介绍了怎样做备份的知识。

20.4.5 口令

我们简单的人用简单的方法。这里有一个简单的方法：每个账号都必须有口令，它必须是不容易猜出来的口令。可惜通过 Internet 发送可重复使用的明文口令已经不再是安全的了，所以如果您允许远程登录系统，就必须使用 SSH 或者其他能保护远程访问安全的系统（在 20.11.3 节讨论）。

20.4.6 警惕性

为了保证您系统的安全，您必须定期地（通常是每天）监视它的健康状况、网络连接、进程表和总体状态。使用本章后面讨论的强大工具来定期做自我评测。安全问题倾向于从小处开始，逐步扩大，所以确定异常越早，解决效果就越好。

20.4.7 普遍原则

有效的系统安全性扎根于常识，下面是一些规则。

- 不要把黑客或好管闲事的职员可能感兴趣的文件放在系统中。商业机密、人员档案、薪酬数据、选举结果等，如果它们在线的话必须小心地处理。用加密的方式保护这样的信息，比起简单地试图防止未经授权的用户访问包含它的文件，可以提供高得多的安全度。
- 站点的安全策略中应该指定如何处理敏感的信息。请参考第 30 章和本章介绍的一些安全标准

www.TopSage.com

（20.3.2 节），获得一些建议。

- 不要为黑客提供可以在系统中造窝的地方。黑客常常攻入一个系统，然后把它用作进入其他系统的根据地。有时候，黑客可能会用您的网络来掩盖自己攻击真实目标的踪迹。能公开访问但确有安全漏洞的服务、任何人都有写权限的匿名 FTP 目录、多人共用的账号以及被遗忘的系统，都是在鼓励黑客的筑巢活动。
- 设置陷阱以检测入侵和入侵企图。像 samhain、**xinetd** 和 John the Ripper 这样的工具（在 20.10 节中介绍）可以帮助您跟得上处理潜在的问题。
- 严密监视由这些安全性工具生成的报告。在一份报告中被忽略的小问题，有可能在下一份报告发出之前变成灭顶之灾。
- 自学有关系统安全的知识。传统的“know-how”、用户培训和常识都是站点安全规划中最重要的部分。把外面的专家请进来，帮助填补空缺，但一定要在您的密切监视和批准之下。
- 四处搜寻，寻找不寻常的活动。调查任何看起来不寻常的事情，比如奇怪的日志消息或某个账号活动的变化（活动频繁，在古怪的时间活动，或者也许在账号主人的休假期间活动）。

20.5 /etc/passwd 和/etc/shadow 文件里的安全问题

口令管理不善是最常见的一种安全弱点。**/etc/passwd** 和**/etc/shadow** 这两个文件的内容决定了谁可以登录，以及一旦他们获准入内可以做什么。这个文件是系统抵抗入侵者的第一道防线。必须小心翼翼地维护它，不能有错误、安全隐患和历史包袱。参考 6.1 节了解有关 **passwd** 文件的更多信息。

在老式系统上，**/etc/passwd** 文件的第二个字段有一个表示用户加密口令的字符串。因为**/etc/passwd** 文件必须对所有人可读，这样像 ls 这样的命令才能工作，所以系统上的所有用户都能读取到加密后的口令字符串。做坏事的人可以加密整个字典，把加密后的结果和**/etc/passwd** 文件里的字符串进行比较。如果匹配到了加密后的字符串，那么就找出了一个口令。

这种威胁会有多大呢？早在 20 世纪 80 年代，至少就有一种方法可以非常快地解密口令[3]，但是普通的黑客只得满足于使用 **crypt** 库例程[4]来加密词典中的单词以便进行比较。20 世纪 80 年代一台“快速的”机器每秒钟可以完成几百次加密。现在使用穷举搜索在 24 小时内就能破解一个 56 位的 DES 密钥。幸亏现代的 UNIX 和 Linux 系统都不使用 **crypt**，因此要安全得多。

这些结果着实令人吃惊，它们表明确实应该限制用户对加密后口令字符串的访问。常用的强制方法是把口令放在一个单独的文件中，这个文件只对 root 可读，而保留**/etc/passwd** 的其余部分不动。于是这个包含实际口令信息的文件被称为隐蔽口令文件（通常是**/etc/shadow**）。所有的现代 Linux 发行版本的默认配置都支持隐蔽口令。

20.5.1 口令的检查和选择

坚持核实（最好是每天）每个登录名是否有口令，这很重要。在**/etc/shadow** 文件中，应该在“daemon”这样的伪用户——它们拥有文件但从不登录——的说明项里，给它们的加密口令域填一个星号（*）或者一个感叹号（!）。这两个字符不与任何口令匹配，这样就可以防止使用这个账号。

/etc/shadow 文件基本上用不着维护。不过下面这个只有一行的 Perl 程序能查找空口令：

[3] 本书作者之一 Evi Nemeth 在 1984 年用一台 HEP 超级计算机，把使用 DES 的 Diffie-Hellman 密钥交换给破解了。虽然 DES 在数学上被认为是安全的，但由于常用的密钥长度短，使其提供的安全性很差。

[4] 不要把 crypt 库例程和 **crypt** 命令弄混了，后者使用另外一种安全性较低的加密方案。

www.TopSage.com

```
$ sudo perl -F: -ane 'print if not $F[1];' /etc/shadow
```

您可以让 **cron** 运行一个脚本来执行这项检查并把结果用邮件发给您。您可以编写一个脚本，用 **diff** 命令把 **passwd** 文件与前一天的版本做比较，并把差异通过电子邮件发给您，帮助核实所有的修改行为是否合法。

20.5.2 口令时限

Linux 的隐蔽口令系统还能让您强迫用户周期性地修改他们的口令，这种工具程序通常称为口令时限。这乍看之下似乎是个好主意，但它有几个问题。用户经常因为不得不修改口令而十分愤慨，由于他们不想忘掉新的口令，所以会选择一些简单的、易于键入和记住的东西。许多用户每次被强迫修改口令时就在两个口令之间来回切换，从而使口令时限失去意义。PAM 模块可以帮助落实安全性强的口令，避免这个问题。

虽然如此，口令还是应该定期修改，特别是 root 口令。root 口令应该很容易在手指上敲出来，这样就可以很快敲进去而不会被看着您的手指在键盘上移动的人猜出来。在我们的站点上，大多数人用 **sudo** 而不是真正的 root 口令，但我们仍然认真地选择 root 口令。有关 **sudo** 的更多信息请参见 3.4.2 节。

chage 程序控制口令的时限。系统管理员使用 **chage** 命令可以设定两次修改口令之间间隔的最长和最短时间，口令作废的日期，在用户口令作废前多少天警告用户，在账号自动锁定之前容许用户多少天不活动等。下面的命令把两次修改口令之间的最少天数设为 2、最多天数设为 90、作废日期为 2007 年 7 月 31 日，并且在接近作废日期的 14 天里警告用户：

```
$ sudo chage -m 2 -M 90 -E 2007-07-31 -W 14 ben
```

20.5.3 组登录名和共享登录名

任何由多人使用的登录名都是坏消息。组登录名（比如“guest”或“demo”）是黑客安家的安全地带，像 HIPAA 这样的联邦法律在很多情况下都会禁止这样做。不要在您的站点上有这类登录名。

同样，不要让用户与家庭或朋友共享登录名。如果小 Johnny 需要一个登录名用于他的科学项目，请给他一个名符其实的登录名。一旦他滥用登录名就可以取消 Johnny 的登录名，这比起取消 Dad 与他共享的账号容易得多，尤其在政府网点上更是如此。

在大多数网点上，“root”是一个组登录名。危险！我们推荐使用 **sudo** 程序来控制对 root 权限的访问。参考 3.4.2 节。

20.5.4 用户的 shell

不要用一个脚本作为无限制（没口令）登录名的 shell。实际上，如果您发现自己需要一个没口令的登录名，那么可能应该考虑代之以使用一个无口令词的 SSH 密钥对。

20.5.5 获得 root 权限的办法

root 登录名唯一的特别之处就在于其 UID 为 0。因为在**/etc/passwd** 文件中可以有多项使用这个 UID，所以以 root 身份登录可以有不止一种途径。

www.TopSage.com

大家网 TopSage.com

黑客一旦获得 root shell，他们安装后门的一种常用方法就是在**/etc/passed** 中编辑新的 root 登录名。像 **who** 和 **w** 这样的程序会指向**/var/run/utmp** 中保存的名称而不是拥有登录 shell 的 UID，因此它们不能暴露出那些看似无辜其实是以 UID 0 登录的黑客。

对这种欺骗的防御方法是编写类似用于发现无口令登录名的那种小脚本。

```
$ perl -F: -ane 'print if not $F[2];' /etc/passwd
```

这个脚本将打印出 **passwd** 文件中 UID 为空（null）或为 0 的所有行。您可以很容易地改写它，找出其组或 UID 同贵单位内部关键人员的组或 UID 相同的可疑项。

还应该检查 **passwd** 中是否有缺少用户名或把标点符号作为用户名的条目。这些条目可能看起来没有意义，但它们常常会让黑客登录。

20.5.6 PAM：验证奇才

PAM，即 Pluggable Authentication Modules（可插入式身份验证模块）最初是由 Sun 公司发明的，Sun 把它作为一种验证用户身份的灵活方法。多年以来，UNIX 环境下的身份验证机制一直就是简单地把用户同他们在**/etc/passwd** 文件里的配置项关联起来而已。后来，因为需要有更强的安全性，而且要支持更大范围的验证机制（像智能卡），所以对灵活性更好的验证方法的需求也就应运而生。某些 PAM LDAP 模块对全局的验证目录集中执行验证。

所有完善的 Linux 发行版本都带 Linux-PAM，它和 Sun 公司 PAM 标准目前的实现没有关系。PAM 的概念很简单：需要验证功能的程序只需要知道有一个模块可以用来替它们执行验证功能就行了。PAM 的设置保证了模块可以随时添加、删除和重新配置——对于编译工具程序的时候就被链接进来（甚至原本就有）的模块来说，没必要这样做。这种结构的效果就是，PAM 已经成为系统管理员极其强大的工具。

PAM 模块是通过**/etc/pam.d** 目录下的文件来进行配置的。这个目录下针对每种服务的文件所包含的配置项都有如下形式：

```
module-type control-flag module-path arguments
```

module-type 字段可以取的值有 auth、account、session 或者 password。auth 确定用户是谁，还可能确定他是哪一个组的成员。account 实行不基于身份验证的决策，比如根据一天中的时间来访问。session 实现了在提供给用户服务的前后需要完成的任务。最后，password 用于要求用户提供验证信息（比如口令）的情况。

control-flag 字段有 4 个可能的取值：required、requisite、sufficient 和 optional。required 和 optional 最常用，它们分别表示：为了程序继续执行模块必须取得成功，或者模块成功与否没有关系。

第 3 个和第 4 个字段是动态加载模块对象的路径和参数。如果路径的第 1 个字符是/，它就被当作是一个绝对路径。否则，这个字段的内容就被追加到默认路径**/lib/security** 的后面。

PAM 是上述口令复杂性难题的解决方案之一。**pam_cracklib** 模块能够强制要求口令符合最低强度要求。具体要求则变化多端，所以要使用 **grep** 找到正确的配置文件。例如，要保证用户口令不能被 John the Ripper 破解（参考 20.10.3 节了解有关它的更多知识），Fedora 上的**/etc/pam.d/system-auth** 应该包括：

```
password required pam_cracklib.so retry=3 minlen=12 difok=4
```

有了这几行，PAM 就会把用户提出的新口令同破解字典和规则集进行对照（这要求有系统库

www.TopSage.com

libcrack，还有一个系统字典**/usr/lib/cracklib_dict.***）。如果用户的口令不符合 **cracklib** 的要求，那么屏幕上就会出现像“The password is too simple（口令太简单）”这样的出错信息。

cracklib 的参数规则很复杂，不过下面是对上述特定配置的解释。

- 参数 `retry=3` 指出用户在输入一个强口令的时候必须输入三遍。
- 参数 `minlen=12` 指出口令的最短长度。大写字母、数字和标点符号由库特殊处理，从而减小了这个最短长度。参数为 `minlen=12` 时，用户能有的最短口令实际为 8 个字符，不是 12 个字符，但是用户设定一个 8 字符口令时必须包括所有 4 种字符类型。
- 参数 `difok=4` 指出新口令至少要有 4 个字符不会出现在老口令中。

现代 Linux 发行版本包含并且默认使用 **pam_cracklib** 模块，但通常不启用口令复杂性规则。

PAM 模块有数十种。您可以从 www.kernel.org/pub/linux/libs/pam 下载专门的模块以及它们的文档。

20.6 POSIX 能力

Linux 系统按照 POSIX 的“能力（capabilities）”概念把 root 账号的权力进行了细分。例如，一种能力允许一个进程在网络套接口和网络接口上执行特权操作，而另一种能力允许一个进程设定和硬件有关的选项。能力可以按照系统化的方式继承、掩码和赠予，程序借助这些方式可以不必获得 root 账号的全部权力就能执行有特权的任务。因为能以比传统的 UNIX 更严格地方式分配特权，所以出现安全漏洞，导致不受限制 root 访问权限泄露的可能性降低了——至少从理论上说是这样。

在实际中间，系统管理员对能力体系的兴趣有限。支持能力的软件不要求在系统管理上专门注意，它只是把自己设为运行在一种更受限的模式下而已。但是大多数 UNIX 和 Linux 软件依然认定了传统的超级用户有全部权力的模型，禁用这个模型会造成很大麻烦，付出很多牺牲。Linux 内核为了保证对传统 UNIX 软件的兼容性所做的一些妥协就是能力体系比它实际应该的更“粗略”[5]。

最后的结果是 POSIX 能力赢得了不错的口碑，但在现实中的使用却很有限。从很大程度上说，它们都被忽略了，但有一个唯一的例外：内核每次执行一个新程序的时候，它都要施加一个“全局能力限制集合”（通过文件**/proc/sys/kernel/cap-bound** 访问和设定）。如果您想要在整个系统内禁用一种或者多种能力，您可以把它们加到这个全局限制集合里。参见 **man capabilities** 的结果了解更多内容。

20.7 setuid 程序

运行 setuid 的程序，特别是运行 setuid 到 root 的程序，很容易导致安全问题。随 Linux 发布的 setuid 命令从理论上讲是安全的，不过，过去已经发现了安全漏洞，将来无疑还会发现漏洞。

把 setuid 问题的数量降到最低的办法，肯定是把 setuid 程序的数量降到最低。在安装需要执行 setuid 的软件包之前请三思，并要避免在自己编写的软件中使用 setuid 功能。虽然 Linux 发行版本各有不同，但一个 Linux 发行版本平均有大约 35 个 setuid 程序。

没有一条规则说 setuid 程序必须以 root 权限运行。如果您需要做的一切就是限制对某个特殊文件或数据库的访问，那么您可以在 **passwd** 文件中添加一个伪用户，**passwd** 文件存在的唯一理由就是拥有受限制的资源。请遵循标准的伪用户约定：使用小的 UID，在口令域中放一个星号，让这个伪用户的主目录为**/dev/null**。

[5] 我们的一位技术审稿人评论道，“实际上，在 Linux 上一直都不能实现 POSIX 能力的继承，这种尴尬还将持续下去。”

www.TopSage.com

您可以在调用 **mount** 命令时使用**-o nosuid** 选项，在个别文件系统上禁止 setuid 和 setgid 的执行。比较好的想法是，把这个选项用在包含用户主目录的目录上，或者在安装从不太可信的管理域来的文件系统时使用。

周期性地扫描硬盘寻找新的 setuid 程序是有用处的。攻破系统安全的黑客有时会创建一个私人 setuid shell 或实用程序来帮助他再次拜访系统。从 20.10 节开始讨论的一些工具能够找出这样的文件，但用 **find** 也一样能做得好。例如：

```
/usr/bin/find / -user root -perm -4000 -print |
    /bin/mail -s "Setuid root files" netadmin
```

将把所有 setuid root 文件的列表邮寄给“netadmin”用户。

20.8 重要文件的权限

Linux 系统中的许多文件必须具有特殊权限，才能避免安全问题。有些发行版本在发售时，其中的软件所具有的权限是为他们自己“友好的”开发环境而设置的。这些权限不一定适合您。

/etc/passwd 和**/etc/group** 不应该让所有人都可写。它们的属主应该是 root，模式应该是 644。**/etc/passwd** 的模式应该是 600——同组人或者其他所有人都没有权限：

```
-rw-------    1 root     root     1835 May 8 08:07 /etc/shadow
```

它的属组应该设置为某些系统组，通常是 root。(**passwd** 命令将运行 setuid，把用户 ID 设置为 root，以便用户不必有**/etc/passwd** 或**/etc/shadow** 的写入权限就可以修改他们的口令。)

以前的 FTP 程序里到处是安全漏洞和不安全的默认配置。FTP 还以明文形式传送身份凭据，这一协议上固有的问题使之无法为当今的 Internet 所接受。

诸如 SFTP 这样的新技术已经取代了 FTP，几乎没有什么情况还应该允许用 FTP 了。不过，在这几种罕见的情况下，也要彻底阅读软件文档，确保配置正确。如果您必须允许匿名 FTP 访问，那要么不准有所有人都能写的目录，要么定期扫描，检查非法的或者敏感的文件。

硬盘分区的设备文件是问题的另一个潜在来源。对某个磁盘设备文件具有读取或写入权限，本质上等同于对它所代表的文件系统中每个文件都具有读取或写入权限。只有 root 应该同时具有读取和写入权限。有时给属组赋予读取权限以便进行备份，但其他人应该没有权限。

20.9 形形色色的安全事务

下面各小节提出了与安全相关的多种主题。它们中的大多数是对管理员有帮助的功能，但如果不加以检查就可能成为给黑客提供“建窝材料”的坏功能。

20.9.1 远程事件日志

系统日志工具（syslog）能把内核和用户进程的日志信息转发给一个文件、一组用户或网络上的另一台主机。请考虑设置一台安全的主机来充当中央日志机器，分析日志文件，以及为感兴趣事件发电子邮件。这项预防措施可以防止黑客通过重写或清除日志文件来抹掉他们的行踪。有关系统日志的更多信息请参见第 10 章。

www.TopSage.com

20.9.2 安全终端

Linux 可以进行配置，限制 root 只能登录特定的“安全”终端。禁止 root 在 SSH、VPN 或者其他远程连接上登录是明智的。网络伪终端也往往设置为禁止 root 登录。

安全通道通常按照配置文件**/etc/securetty** 中的一张 TTY 设备列表来设置。还有可能用**/etc/security/access.conf** 文件中的配置项把非 root 用户限制在特定位置登录，或者用**/etc/security/time.conf** 文件中的配置项限制他们在特定时间登录。

系统管理员仍然能正常登录，使用 sudo 获得对超级用户的权限的。

20.9.3 /etc/hosts.equiv 和~/.rhosts

hosts.equiv 和**~/.rhosts** 文件定义的主机在管理上是彼此“等价的”，用户无需键入口令即可（通过 **rlogin**）登录和（通过 **rcp**）复制文件。在 UNIX 和谐和睦的时代，这个工具程序一度得到普遍使用，但每个人最后都感到它非常讨厌，并且认识到这可不是个好主意。

幸好 SSH 协议实际上消除了客户使用不安全的等效程序 **telnet**、**rsh** 和 **rlogin**。它的用法在本章后面介绍。

rlogin 的一些替代软件（包括 SSH!）如果配置不正确的话，它们会注意**.rhosts** 和**/etc/hosts.equiv** 内的配置。为了增加安全性，您可以为每位用户（包括 root）创建**/etc/hosts.equiv** 文件和一个不可写的、长度为 0 的**~/.rhosts** 文件。如果存在一个文件并且没有被人碰过，那么估计它在凌晨 3:00 时的状态总要比估计一个不存在的文件的状态容易一些。当您在跟踪入侵者和他们攻破系统的企图时，这一点区别是至关重要的。

20.9.4 安全与 NIS

除了用作本节的标题之外，这两个名词永远不应该一起使用。NIS（Network Information Service，网络信息服务，以前叫做黄页）是一种 Sun 的数据库发布工具，许多站点用它来保存和发布像**/etc/group**、**/etc/passwd** 和**/etc/hosts** 这样的文件。很遗憾，其“易于信息访问”的性质使它成为吸引黑客的诱饵。

更安全和可靠地发布这些文件的一种方法是创建像“netadmin”这样的一个登录名，并把这些文件最新的副本放在目录**~netadmin** 下。然后您可以在每台客户机上运行一个 **cron** 调用的脚本来用 **scp** 复制文件、检查完整性和安装文件。**scp** 是 SSH 的一部分，参考 20.11.3 节了解有关 SSH 的更多知识。

20.9.5 安全与 NFS

NFSv4 是 IETF 对 Sun 公司以前的协议所做的一种扩展，它具有很强的安全性，而且有其他一些比早先的实现更好的优点。虽然该协议尚不完善，但是开发工作开展顺利，所以在 Linux 2.6 内核中包含了 NFSv4。

NFS 的老版本都使用一种安全性很弱的模型。参考 16.1.7 节了解更多有关 NFS 安全的知识。使用 **showmount -e** 可以看到哪些文件系统被导出了，导出给谁了。每个被导出的文件系统都应该有一个访问列表，并且所有的主机名都应该用全名。

www.TopSage.com

20.9.6 安全与 sendmail

sendmail 是一个庞大的程序，它要以 root 权限运行（至少原来是这样）。因此，**sendmail** 常常成为黑客攻击的目标。要确保在所有系统上运行的都是最新版本的 **sendmail**。因为安全问题是发行新软件版本最可能的原因之一，所以，除了最新版本外，所有版本的 **sendmail** 都可能有问题，有关 sendmail 的更多信息请参见第 19 章。

专门和 **sendmail** 安全性有关的细节在 18.12 节介绍。您可以从 www.sendmail.org 获知特定版本的信息。

20.9.7 安全与备份

定期的系统备份是任何网点安全计划的一个重要组成部分。要保证所有分区都被定期地转储到磁带上了，而且还要在异地保存一些备份。如果发生重大的安全事故，您可以恢复到一个未受污染的检验点。有关备份的更多信息请参见第 9 章。

备份也可以成为一种安全风险。因为一旦把磁带安装到驱动器上，任何人都可以读取它的内容，所以您必须妥善严密地保管所有的备份磁带。备份时考虑使用加密。如果您在考虑签订合同使用一种存储设施，那么先要求实际试用一下。

20.9.8 病毒与蠕虫

Linux 一直都最不容易感染病毒。Linux 上只有很少的一些病毒（其中大部分的本质还是学术研究性质的），而且没有哪种病毒对 Linux 造成了 Windows 上司空见惯的那种损失。尽管如此，仍然没有阻止某些防病毒厂商去预测 Linux 平台将被恶意软件所摧毁——当然，除非您以一个特殊的指导价，购买这些商家的防病毒产品。

Linux 上少有恶意软件的真正原因还不清楚。有些人认为 Linux 只是因为比其桌面竞争对手的市场份额低，因此没有成为病毒制造者感兴趣的目标。其他人则坚持认为 Linux 受访问控制的环境限制了自行传播的蠕虫或者病毒的扩散破坏。

后一种观点有一定道理。因为 Linux 把系统中可执行程序的写权限限制在文件系统一级，没有特权的用户账号不能影响到环境的其他地方。除非病毒代码是以 root 身份运行的，否则其作用范围就会大大地受到限制。于是最关键的一点就是不要用 root 账号做日常的工作。

或许与人们的直觉相反的是，在 Linux 服务器上运行防病毒软件有一个正确的理由，即保护您本地的 Windows 系统免于受到专门针对 Windows 的病毒的侵害。邮件服务器可以扫描电子邮件中的附件有没有携带病毒，文件服务器能够扫描共享文件看是否受到病毒感染。不过，这些解决方案应该作为对桌面机器防病毒保护的一种补充，而不能代替后者。

Tomasz Kojm 开发的 ClamAV 是一种用于 Linux 的流行的免费防病毒产品。这种得到广泛使用的 GPL 软件是一套完整的防病毒工具库，带有成千上万种病毒的特征。您可以从 www.clamav.net 下载到最新版的 ClamAV。

20.9.9 特洛伊木马

特洛伊木马是一种表里不一的程序。特洛伊木马的一个例子是名为 turkey 的程序，很久以前在 Usenet 流传。这个程序说它将在您的终端屏幕上绘制一幅火鸡图片，但实际上它会删除您主目录下的

www.TopSage.com

大家网 TopSage.com

文件。

另一个引起更大争议的特洛伊木马的例子是，2004 年和 2005 年索尼公司发布的许多音乐光盘中带的保护版权的软件。这种指导思想上就有错的行为的目的是阻挠共享音乐的人，索尼公司在未征得听音乐的人同意的情况下，在 Windows 系统上安装了保护软件。这个软件给电脑主机引来了可以被蠕虫和病毒利用的安全漏洞。

特洛伊木马的不完整碎片不时会出现在主要的 Linux 软件包中。**sendmail**、**tcpdump**、OpenSSH 和 InterBase 都发布了有关其产品中出现恶意软件的安全建议。这些特洛伊木马一般植入恶意代码，让攻击者能随心所欲地访问受害者的系统。还好，大多数软件商都会在一两周内纠正软件，并发出安全建议。对于 Linux 主机上运行的任何网络软件包，都要关注其安全邮递列表。

在列出 Linux 界过去几年中发现的与安全有关的异常行为的数量后发现，引起人们注意的是特洛伊木马的事故是如此之少。这种情况在很大程度上要归功于 Internet 通信的速度。明显的安全性问题往往很快被发现并广泛讨论。在知名的 Internet 服务器上，恶意的软件包不会被保留太久。

可以肯定的是，任何被发现怀有恶意的软件都会在 Internet 上得到广泛讨论。如果您希望在安装软件之前快速确认一下，可以用这个包的名字在您最喜欢的搜索引擎上搜索一下。

20.9.10 Rootkits

最熟练的黑客总是会试图掩盖他们的踪迹，避免自己被发现。他们经常希望能连续利用您的系统非法扩散软件、探测其他网络或者发起针对其他系统的攻击。他们经常使用“rootkit”来帮助自己不会被检测到。索尼公司的特洛伊木马就利用了类似 rootkit 的功能来防止自己被用户发现。

rootkit 是隐藏重要系统信息（如进程、硬盘或者网络活动）的程序和补丁。它们有许多种风格，复杂程度也千变万化，简单到替换应用程序（如被黑过的 **ls** 和 **ps** 版本），复杂到几乎不可能被发现的内核模块。

基于主机的入侵检测软件，如 samhain（下面会介绍）是一种监视系统是否有 rootkit 出现的有效手段。虽然有程序能帮助系统管理员把 rootkit 从一个被感染的系统中删除，但是考虑到做一次彻底的清理工作所花费的时间，更节省时间的做法是保存数据、重新格式化硬盘，从头开始装系统。最高级的 rootkit 知道常见的删除程序，会试图破坏它们。

20.10 加强安全的工具

前面几节中提到的一些费时间的工作可以用免费获得的工具自动完成。下面是您想看看的几个工具。

20.10.1 Nmap：扫描网络端口

nmap 是一个网络端口扫描程序。其主要功能是检查一组目标主机，看看在哪个 TCP 和 UDP 端口上有服务程序在监听[6]。因为大多数网络服务都与“众所周知的”端口号相关联，所以这项信息会告诉您有关一台机器上正在运行的软件的很多信息。

运行 **nmap** 是一种很好的方法，可以弄清楚系统在打算搞破坏的人看来是个什么样子。例如，下

[6] 正如在第 12 章中介绍的那样，一个端口就是一个有编号的通信通道。一个 IP 地址标识一整台机器，IP 地址+端口号则标识出那台机器上的一个特定服务程序或网络通信对话。

www.TopSage.com

面是来自一台 RHEL 默认安装机器的报告：

```
$ nmap -sT rhel.booklab.example.com

Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-15 23:48 EDT
Interesting ports on rhel.booklab.example.com (192.168.1.31):
(The 1668 ports scanned but not shown below are in state: closed)
PORT    STATE  SERVICE
22/tcp  open   ssh
25/tcp  open   smtp
111/tcp open   rpcbind
631/tcp open   ipp

Nmap run completed -- 1 IP address (1 host up) scanned in .158 second
```

在默认情况下，**nmap** 带的**-sT** 参数要求以正常方法尝试连接目标主机上的每个 TCP 端口[7]。一旦建立连接，**nmap** 将立即断开，这样做虽然不礼貌，但对一个正常编写的网络服务程序来说是无害的。

从上面的例子我们可以看出，主机 rhel 正在运行几个可能没用到的服务程序，这几个程序在历史上就与安全性问题息息相关：**portmapd**（`rpcbind`），**CUPS**（`ipp`），可能还包括 **sendmail**（`smtp`）。几条潜在的攻击路线已经相当清楚了。

nmap 输出中的 `STATE` 一栏显示“`open`”表示端口有服务程序，“`unfiltered`”表示端口没有服务程序，“`filtered`”表示因为防火墙的干扰而无法探测到该端口。unfiltered 的端口是最典型的情况，除非 **nmap** 在执行一次 ACK 扫描，否则通常都不显示它们。例如，下面的输出来自一台更安全的服务器：

```
$ nmap -sT secure.example.com

Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-15 23:55 EDT
Interesting ports on secure.example.com (192.168.1.33):
(The 1670 ports scanned but not shown below are in state: closed)
PORT     STATE  SERVICE
22/tcp   open   ssh
1241/tcp open   nessus

Nmap finished: 1 IP address (1 host up) scanned in 0.143 seconds
```

在这个例子中，很明显，主机被设置成只运行 SSH 和 Nessus 网络扫描程序。防火墙封锁了对其他端口的访问。

除了直截了当的 TCP 和 UDP 探测之外，**nmap** 还有一整套悄悄探测而不发起实际连接的方法。在大多数情况下，这些探测方法会发送看上去像是出自 TCP 会话中间（而不是开始）的包，然后等待诊断包被送回。这些秘密的探测方法也许能有效地绕过防火墙或避免被寻找端口扫描程序的网络安全监视器检测到。如果您的站点使用了防火墙（参见 20.12 节），用这些可供选择的扫描模式来探测它，看看它们能找到什么是个好主意。

nmap 有一种神奇而有用的能力：通过查看一个远程系统 TCP/IP 实现的细节来推测该系统运行的是什么操作系统。它有时候甚至能确定一个开放端口上正在运行什么软件。**-O** 和**-sV** 选项分别打开这些功能。例如：

```
$ nmap -O -sV secure.example.com

Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-16 00:01 EDT
Interesting ports on secure.example.com (192.168.1.33):
(The 1670 ports scanned but not shown below are in state: closed)
```

[7] 实际上，默认情况下只检查特权端口（端口号小于 1024 的端口）和众所周知的端口。请使用**-p** 选项明确地指定要扫描的端口范围。

www.TopSage.com

```
PORT     STATE  SERVICE VERSION
22/tcp   open   ssh     OpenSSH 3.6.1p2 (protocol 2.0)
1241/tcp open   ssl     Nessus security scanner
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.5.25 - 2.6.8 or Gentoo 1.2 Linux 2.4.19
    rc1-rc7, Linux 2.6.3 - 2.6.10
Nmap finished: 1 IP address (1 host up) scanned in 8.095 seconds
```

这项功能对获取本地网络的详细目录非常有用。很遗憾，它对黑客也非常有用，他们可以把攻击集中在目标操作系统已知的弱点上。

也请记住，大多数系统管理员通常不喜欢您扫描他们的网络并指出网络的薄弱之处，不论您的动机有多友好。在没有得到其他网络的系统管理员同意之前，决不要用 **nmap** 来扫描别人的网络。

20.10.2 Nessus：下一代的网络扫描程序

Renaud Deraison 在 1998 年发布了一个名为 Nessus 的软件包，它是一种强大且有用的安全漏洞扫描程序。在编写本书（英文版）的时候，它使用了超过 10 000 个插件来检查本地和远端的安全漏洞。虽然现在这个软件的源代码没有公开，而是一种专有的产品，但是它可以自由使用，而且定期会发布新插件。它是使用最广、最完整的安全漏洞扫描程序。

Nessus 称自己是不会对任何东西想当然的安全扫描器。例如，它不会假定所有的 Web 服务器都运行在 80 端口，而是在任意端口上扫描检测 Web 服务器，然后再检查它们有没有安全漏洞。它不使用它所连接的服务报告出来的版本号，而是自己尝试利用已知的漏洞来看服务是否会受到影响。

虽然第一次让 Nessus 在您的系统机器上运行要花大量的安装时间（它需要的大量软件包，一般发行版本默认是不安装的），但它还是值得一试的。

Nessus 系统包括一个客户端和一个服务器端。服务器充当了一个数据库，而客户端负责处理 GUI 显示。您必须在一个 UNIX 或者 Linux 系统上运行服务器，但有各种各样的客户端，能在其他平台上控制和显示 Nessus。

Nessus 最大的优点之一就是它采用了模块化的设计，这让第三方很容易就可以增添新的安全检测功能。幸亏有一个积极的用户群（尽管最近的许可证变了），Nessus 可能是未来几年中一种很有用的工具。

20.10.3 John the Ripper：找出不安全的口令

避免选择不良口令的一种方法就是您自己尝试攻破口令，迫使您已经攻破其口令的用户修改他们的口令。John the Ripper 是由 Solar Designer 提供的一种高级工具，它在一种工具里实现了各种破解口令的算法。它代替了老程序 **crack**，后者在本书的上一版中介绍过。

即使绝大多数系统使用隐蔽口令文件把加密后的口令隐藏起来避免公众查看，验证用户的口令是否能抵抗得住破解仍然是明智的。了解用户的口令会很有用，因为人们往往反复使用相同的口令。一个口令可能提供对另一个系统的访问权，可以解密用户主目录下保存的文件，允许访问 Web 上的资金账号（不用说，这样重复使用一个口令对安全性来说可不大聪明。可是没有人想记住 10 个不同的口令）。

即使 John the Rip 的内部很复杂，但是这个程序使用起来却极其简单。直接对要破解的文件（最常用的就是**/etc/shadow**）执行 **john** 命令，然后看看有什么神奇的事情发生：

www.TopSage.com

```
$ sudo ./john /etc/shadow
Loaded 25 password hashes with 25 different salts (FreeBSD MD5 [32/32])
password  (bad2)
badpass   (bad1)
```

在这个例子中，从隐蔽文件里读出了 25 个不同的口令。随着口令被破解，John 会把它们显示在屏幕上，并且把它们保存在一个叫做 **john.pot** 的文件里。输出的左边一列是口令，右边一列的括号里是登录名。在 **john** 执行完以后再次显示口令，可以带**-show** 参数执行相同的命令。

在写本书(英文版)的时候，John the Ripper 的最新版本是 1.7.0.2。它可以从 www.openwall.com/john 获得。因为 John the Ripper 的输出中包含它已经攻破的口令，所以您应该小心地保护它，并在做完之后立即删除。

20.10.4 hosts_acces：主机访问控制

网络防火墙是阻挡非授权主机访问网络的第一道防护，但是它们不应该仅有一道屏障。Linux 使用了两个文件**/etc/host.allow** 和**/etc/hosts.deny**，根据网络请求的来源限制对服务的访问。**host.allow** 文件列出了允许连接到一个特定服务的主机，而 **hosts.deny** 文件则负责限制访问。不过，这两个文件只控制对有 hosts_access 功能的服务（如 **xinetd** 所管理的那些服务、**sshd** 和某些配置的 **sendmail**）的访问。

在大多数情况下，明智的做法是先做限制，然后只允许从指定主机访问关键服务。我们建议，默认在 **hosts.deny** 文件中加上下面这一行配置，拒绝所有的访问：

```
ALL:ALL
```

接下来，您可以在 **hosts.allow** 文件中逐个开放访问许可。下面的配置允许从网络 192.168/16 访问 SSH，而从任何地方访问 **sendmail**。

```
sshd: 192.168.0.0/255.255.0.0
sendmail: ALL
```

两个文件每行配置的格式都是 *service: host* 或者 *service: network*。失败的连接企图被记录到 syslog 中。从没有得到允许访问该服务的主机来的连接会被立即关闭。

大多数 Linux 发行版本默认都带 **host.allow** 和 **hosts.deny**，但是它们通常为空。

20.10.5 Samhain：基于主机的入侵检测

Samhain 既是一个基于主机的入侵检测系统，也是一个文件完整性的监视程序，它由 Samhain Lab（la-samhna.de）的 Rainer Wichmann 负责开发和维护。多亏它的模块化接口，samhain 能够监视文件的完整性、检查挂载的文件系统、搜索 rootkit 等。例如，smahain 很容易就能判断出一个入侵者用一个能够把口令记在一个秘密文件中的 **login** 替换了您的那个**/bin/login**。

Samhain 以控制台为中心的设计能够确保把日志数据记录到一台可信的主机上，从而有助于维护审计数据的完整性。从 Web 界面或者通过分发的电子邮件能够看到主机变化的报告。Samhain 的日志可以用作取证工具，帮助重构一起安全事故过程中发生的一系列事件。

Samhain 对照数据库来检查主机的特性，该数据库包含在数据库建立之时已知是正常的文件信息和校验和。其整体思想为：从系统的一个可信状态构建基准数据库，然后定期对比检查和历史数据库的差异。已知在正常情况下做修改的文件可以配置不产生报警。当一个系统的配置发生变化，或者安

www.TopSage.com

装了一个新软件的时候，应该重建数据库，从而避免真正的问题淹没在大量的虚假报警中。

整个 samhain 系统包括 3 个组成部分。在客户端运行的 **samhain** 代理（agent），它向日志（log）服务器报告数据。日志服务守护进程叫做 **yule**，接受从 **samhain** 客户端来的连接，把数据加入到配置好的日志设施里，通常是 **syslog** 或者一个数据库。基于 Web 的管理控制台叫做 Beltane，它管理系统的基准数据库、编辑配置，并且用一个数字签名签发配置和基准数据库。

该系统基于组件的设计具有比传统的文件完整性扫描工具更好的几个优势。采集到的数据都放在一台可信的主机上，所以攻击者破坏数据库完整性的机会比较小。由于在一个重要位置采集信息，所以管理起来也更容易了。

您可以把 samhain 配置得非常偏执。诸如把自己隐藏在进程列表里这样偷偷摸摸的事情，samhain 都可以做到。samhain 还能把它的数据库追加在一个文件（如 JPEG 文件）之后，而这个文件也不会因为多加的数据而被损坏，从而能够把数据库隐藏在一张图像里。

用 Web 界面可以让创建的配置文件和数据库有加密的签名，这有助于检查到对这些文件未经授权的访问。

下面是一个 samhain 给 **syslog** 发的报告的例子：

```
# CRIT   : [2006-06-25T19:31:48-0600] msg=<POLICY [ReadOnly] --------T->,
    path=</bin/login>, ctime_old=<[2006-06-26T01:24:34]>, ctime_new=<[2006-
    06-26T01:31:47]>, mtime_old=<[2006-06-26T01:24:34]>, mtime_new=<[2006-
    06-26T01:31:47]>,
    48CA06CC50B857DE77C27956ADE7245B0DF63F6A8A42F5B7
```

看上去有点神秘，但是这条日志表明有一则严重性为 CRITICAL 的日志消息出现，/bin/login 这个程序的 ctime 和 mtime 属性都改变了。具体时间待调查。

遗憾的是，维护 samhain 需要大量的开销。这个基于和系统基准对比的系统，和任何类似的系统一样，都会因为日常的合法活动而产生大量的误报。Samhain 的配置文件可以帮助清理频繁变化的文件产生的误报，但是系统管理员不能预料到一切。要牢记一点，总是要付出相当的工作量，特别是对于大规模的部署来说更是如此。

20.10.6 SELinux

正如我们已经看到的那样，Linux 的安全模型有它自己的缺点。自主访问控制（这个概念的含义是，基于账号的权限来允许其对文件的访问）是一种方便但不安全的控制对文件系统对象进行访问的方法。它从本质上说是基于信任的：信任有访问权限的用户不是恶意的，信任系统管理员知道一个软件包中每个文件的正确权限，信任第三方软件包有到位的强大控制力来安装它们自己。但即便所有这些都做到了，一个软件上的安全漏洞仍然会让系统变得毫无保护。

SELinux 通过使用强制访问控制（也叫做 MAC）解决了这个问题。在 MAC 下，用户没有对对象的访问控制权。相反，系统管理员规定了系统级的访问策略。MAC 策略要落实得好，就得依靠最小特权原则（只有必须的时候才允许访问），这非常像一个正确设计好的防火墙，只让认得出的特殊服务和客户机通过。MAC 能够让在执行代码上有漏洞的软件（例如，缓冲区溢出）不能破坏系统，办法是把漏洞的作用范围限制在该软件所需要的特定资源上面。

SELinux 是一个 NSA 的项目，从 2000 年末就可以免费获得。它已经被集成到 2.6 版的 Linux 内核中。不过，各个 Linux 发行版本采用的 SELinux 相当薄弱，而 RHEL 和 Fedora 是个例外。

访问控制策略的开发是一个复杂的话题。至少有一家公司能提供一种 3 天的 SELinux 策略。例如，为了保护一个新的守护进程，一个策略必须仔细枚举出所有的文件、目录和该进程需要访问的其他对象。对于像 **sendmail** 或者 Apache **httpd** 这样的复杂软件，这个任务也非常复杂。

www.TopSage.com

幸好网上有很多通用策略。这些策略安装起来很容易，而且可以针对您的特殊环境进行配置。在 seedit.sourceforge.net 可以找到一个完善的策略编辑器，其目标是方便策略应用的使用。



SELinux 出现在 RHELv4 以后的版本里。Fedora 把该软件收入 Core 2 版，并且在 Core 3 里加入了更多的支持。默认安装的 Fedora 和 RHEL 实际上一装好就具有一些 SELinux 的保护功能。

文件**/etc/selinux/config** 控制着 SELinux 的配置。其有意思的配置行如下：

```
SELINUX=enforcing
SELINUXTYPE=targeted
```

第一行有三个可能的值：`enforcing`、`permissive` 或者 `disabled`。设置 `enforcing` 则保证会实施被加载的策略，禁止出现违反该策略的情况。设置 `permissive` 则允许违反策略，但会通过 syslog 把它们记录下来。设置 `disabled` 则完全关闭 SELinux。

`SELINUXTYPE` 指要应用的策略类型。Red Hat 和 Fedora 都有两种策略：`targeted` 为 Red Hat 已经保护起来的守护进程规定了更多安全措施[8]，`strict` 则保护整个系统。虽然有 strict 这个策略，但是 Red Hat 并不支持它，这个限制太严格了，系统很难使用。策略 `targeted` 提供了对重要的网络守护进程的保护，又不会影响到整体系统的使用，至少从理论上说是这样。但即便是 `targeted` 策略也不完美。如果您在新安装的软件上遇到了问题，可以检查**/var/log/messages** 看看有没有 SELinux 的错误。



SUSE 使用它自己的 MAC 实现，这个实现称为 AppArmor，它不带 SELinux。



Russel Coker 维护着 Debian 和 Ubuntu 能用的 SELinux 软件包，他是 Red Hat 公司的人，是他做的 `strict` 和 `targeted` 这两个策略。

20.11 加密的安全工具

Linux 系统上大多数常用的网络协议都可以追溯到 Internet 和现代加密算法的广泛使用之前。安全性并不是许多协议设计时的一个考虑因素；而在其他协议中，只是传送纯文本的口令，或者随便检查一下数据包是否来自受信任的主机或端口，从而把对安全性的考虑抛到了一边。

现在这些协议发现自己是在大型企业的 LAN 和 Internet 的大江大浪中运行，在这些环境下险象环生，但必须认识到，所有流量都是敞开的，随便可以监听到。不仅如此，几乎也没办法防止别人主动干扰网络会话。您怎么才可以相信自己真正在和谁谈话？

密码为许多这样的问题提供了一条解决之道。很长时间以来，已经有可能把消息打乱使得偷听者不能解密，但这只是密码奇迹的开始。公钥加密法和安全散列法的发展已经使得加密系统的设计几乎能够满足任何想象得到的需求。

遗憾的是，这些数学上的发展在转化为安全而有用的、能被广泛接受和了解的软件时大多失败了。加密软件系统的开发者往往对可证明的正确性和绝对的安全性非常感兴趣，而对一个系统是否对真实世界有实际意义不那么感兴趣。目前的大多数软件往往做得过头了，所以在给用户些许机会时他们却尖叫着逃走也就不奇怪。现在，使用加密法的人大部分是对加密法感兴趣的计算机业余爱好者、有关

[8] 这些受到保护的守护进程有 **httpd、dhcpd、mailman、named、portmap、nscd、ntpd、mysqld、postgres、squid、winbindd 和 ypbind。**

www.TopSage.com

加密的理论学家和那些因为管理策略而别无选择的人。

我们在今后的几年有可能看到一条适合加密技术发展的道路，也可能看不到。在此期间，下面几小节将讨论一些目前能提供的方法。

20.11.1 Kerberos：用于网络安全的统一方法

MIT（麻省理工学院）设计的 Kerberos 系统试图用一种统一和可扩展的方式处理某些网络安全问题。Kerberos 是一种验证系统，一个“保证”用户和服务名符其实的工具。它不提供此外任何附加的安全性或加密。

Kerberos 使用 DES 来构造嵌套的凭证（credential）集，称为“标签（ticket）”。标签随着网络传递以证明您的身份，并为您提供对网络服务的访问权。每个 Kerberos 站点必须保证至少有一台物理上安全的机器（称为验证服务器）运行 Kerberos 守护进程。这个守护进程根据请求验证身份的用户或所提供的凭证（比如口令）给它们发放标签。

从本质上说，Kerberos 只从两个方面改进了传统的 Linux 口令的安全性：它从不在网络上传送未加密的口令，并且它使用户不需要再反复键入口令，这就使得保护网络服务的口令更为容易。

Kerberos 界夸奖说，Bill Bryant 的“Designing an Authentication System: a Dialogue in Four Scenes”是已出的有关密码系统的文档中最透彻、最令人愉快的一种。它是任何对密码学感兴趣的人的必读资料，从 web.mit.edu/kerberos/www/dialogue.html 可以得到它。

Kerberos 比“彻底忽略网络安全性”模型提供了更好的网络安全性。不过，它既不彻底安全，安装和运行起来也不是非常方便。它不能代替本章中介绍的其他任何安全措施。依我们看，大多数站点没有它会更好。好的系统维护管理和用于远程登录的集中加密解决方案（比如 SSH，参见 20.11.3 节），应该能为用户提供更充分的安全水平。

遗憾的是（而且或许可以说是不出人们所料），作为 Windows 一部分而发布的 Kerberos 系统使用了专有的、没有文档的扩展。结果它不能和基于 MIT 代码的版本实现互操作了。

20.11.2 PGP：很好的私密性

Philip Zimmermann 的 PGP（Pretty Good Privacy）包提供了一套由基本加密实用程序组成的工具，其重点放在电子邮件的安全上。它可用于加密数据、生成签名以及验证文件和消息的来源。

规整或停止 PGP 发布的努力使得它的历史相当曲折。McAfee 在其 E-Business 服务器产品中包括了 PGP，PGP corporation 将 PGP 用于各种各样的加密产品。GNU 项目在 www.gnupg.org 上提供了一种优秀、免费且使用广泛的实现，叫做 GnuPG。经过美国政府审查的 PGP 版本可在美国使用，而从 www.pgpi.org 可获得一个更强大、有更多不同加密方法的国际版本。国际性的存档站点似乎并没有筛选出美国地址，所以美国用户必须非常小心，不要不留神去 www.pgpi.org 下载了功能完整的 PGP 版本。

PGP 是常用加密软件中最流行的。很遗憾，UNIX/Linux 上的版本是如此的严丝合缝，以致于您必须了解相当多的加密背景知识才能使用它。幸运的是，PGP 带有一份 88 页的加密学论文，能对设置提供帮助。虽然您可能发现 PGP 对自己的工作有用，但我们不建议您为了用户而支持它，因为已经知道它会引起许多奇怪的问题。我们已经找到了 PGP 的 Windows 版本，它比带 38 种不同运行模式的 Linux **pgp** 命令要容易使用得多。

Internet 上的软件包在发布时常常带有一个 PGP 签名文件，它的用处是保证软件的来源和纯正。遗憾的是，不是坚持使用 PGP 的用户很难验证这些签名的有效性——不是因为验证过程复杂，而是因为必须从已经直接验证过身份的人那里收集一个您个人的公钥库，才能享受到真正的 PGP 安全作用。

www.TopSage.com

下载一个公钥和一个签名文件以及软件分发版本，其安全性与单独下载软件的发行版本大致相当。

20.11.3 SSH：安全的 shell

由 Tatu Ylönen 编写的 SSH（secure shell）系统，是 **rlogin**、**rcp** 和 **telnet** 的有安全保障的替代软件。它使用加密验证来确认用户的身份，并对两台主机之间的所有通信加密。SSH 所使用的协议在设计上能经受各种潜在的攻击。该协议的文档是 RFC 4250 到 4256，现在由 IETF 确定为建议标准。

SSH 已经从一个免费发布的开放源码项目（SSH1）转变为一个商业产品，它使用一种稍有不同（并且更安全）的协议（SSH2）。幸好开发源代码界已经用发布优秀的 OpenSSH 软件（由 FreeBSD 维护）作出了回应，后者现在已经实现了这两种协议。

SSH 的主要组件是一个服务器守护进程 **sshd** 和两个用户级的命令：用于远程登录的 **ssh** 和用于复制文件的 **scp**。其他组件有：生成公钥对的 **ssh-keygen** 命令，两个帮助支持加密 X Windows 的实用程序。

sshd 可以用几种不同方法验证用户的登录。由作为管理员的您来决定接受这些方法中的哪一种。

- **方法 A**：如果在**~/.rhosts**、**~/.shosts**、**/etc/hosts.equiv** 或**/etc/shosts.equiv** 中列出了用户从远程登录的主机名，那么用户无需检查口令就可以登录。这个方案仿照了旧的 **rlogin** 守护进程，依我们看正常使用不能接受这种方法。
- **方法 B**：作为方法 A 的改进，**sshd** 还可以使用公钥加密法来验证远程主机的身份。为了做到这一点，远程主机的公钥（在安装时生成）必须在本地主机的**/etc/ssh_known_hosts** 文件或用户的**~/.ssh/known_hosts** 文件中列出。如果远程主机可以证明它知道对应的私钥（通常保存在 **/etc/ssh_host_key** 里，这是一个所有人都不可读的文件），那么不必询问用户口令就可以让他们登录。方法 B 比方法 A 的限制更多，但我们认为它仍然不够安全。如果发起方主机的安全性受到损害，那么本地站点也会被连累。
- **方法 C**：**sshd** 可以使用公钥加密法来建立用户的身份。在登录时，用户必须能够访问他或她的私钥文件的副本，而且必须提供一个口令来解开它。这种方法是最安全的，但它设置起来有点烦人。它还意味着您不能在旅行时登录，除非随身带着私钥文件的副本（也许是在便携机中）。如果您决定使用密钥对，那么在排除故障期间要扩充使用 **ssh -v**。
- **方法 D**：最后，**sshd** 可以简单地允许用户输入她或他的标准登录口令。这使得 **ssh** 的表现非常像 **telnet**，除了口令和会话都被加密了之外。这种方法的主要缺点是系统登录口令相对脆弱（常常被限制为 8 个有效字符），并且有专为攻破它们而设计的现成工具（比如 John the Ripper）。不过，这种方法可能是正常使用的最佳选择。

身份验证规则在**/etc/sshd_config** 文件中进行设置。您会立刻发现这个文件已经充满了配置垃圾，不过它们中的大多数确实可以忽略，不会有问题。与身份验证有关的选项在表 20.1 中显示。

表 20.1 /etc/sshd_config 中同身份验证有关的选项

选 项	方 法[a]	默 认	打开时的含义
RhostsAuthentication	A	无	允许通过**~/.shosts**、**/etc/shosts.equiv** 等文件登录
RhostsRSAAuthentication	B	有	允许**~/.shosts** 等文件，但是还需要主机密钥
IgnoreRhosts	A、B	无	忽略**~/.rhosts** 和 **hosts.equiv** 文件[b]
IgnoreRootRhosts	A、B	无[c]	禁止对 root 进行 **rhosts/shosts** 验证
RSAAuthentication	C	有	允许对用户逐个进行密钥加密验证
PasswordAuthentication	D	有	允许使用标准的登录口令

a. 此变量关联的认证方法。

b. 但 honor**~/.shosts** 和 **shosts.equiv** 仍然有效。

c. 默认值为 IgnoreRhosts。

www.TopSage.com

我们的建议配置如下，它允许方法 C 和 D 但不允许方法 A 或 B：

```
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication yes
```

让 root 远程登录始终不是明智之举。应该通过使用 **sudo** 获得超级用户访问权限。要鼓励这种做法，可使用下面这个选项：

```
PermitRootLogin no
```

20.11.4 一次性口令

像 John the Ripper 这样的穷举破解工具引起了人们对静态口令不安全性的关注。即使采取了像 PAM 和口令时限这样的措施，用户仍然会选择容易记住但很脆弱的口令，而且会把口令写在一次性贴纸上，和同事们共用。一次性口令要强制每次登录使用唯一口令来解决这个问题。因为口令总是在变，所以穷举破解工具就没有用处了。

现如今，一次性口令最常出现在商业性的安全产品中。一些厂家提供的一次性口令系统采用了小“钥匙扣”（或者信用卡）大小的设备，这些设备有 LCD 显示，能随时为用户的登录生成口令。分发和替换这些小硬件设备至少也会让您的系统管理助理每周忙上几个小时。

20.11.5 Stunnel

Michal Trojnara 编写的 stunnel 是一个开放源代码的软件包，stunnel 能加密任何 TCP 连接。这个软件使用 SSL（Secure Sockets Layer，安全套接口层）创建一条端到端“隧道”，通过这条隧道，stunnel 能向一个未加密的服务传送数据，或者把数据从那里传回来。现在知道该软件能和 Telnet、IMAP 和 POP 这样无安全措施的服务配合得很好。

在客户端和服务器端都要运行一个 stunnel 守护进程。本地的 stunnel 通常会接受在服务的传统端口（例如，SMTP 的 25 端口）上出现的连接，把连接通过 SSL 送到一条通往远端主机的 stunnel 隧道里。远端的 stunnel 接受这条连接，解密传来的数据，把数据送给该服务器正在监听的远程端口。这个系统能让没有加密功能的服务用上加密所带来的机密性和完整性，又不要求对软件做任何修改。只需要配置客户端软件找到本地系统上的服务，而在最终提供服务的服务器端不用配置。

Telnet 是个很好例子，因为它只包括一个监听单个端口的守护进程。要把 Telnet 连接送入 stunnel 隧道，第一步是创建一个 SSL 证书。Stunnel 与 SSL 库无关，所以基于标准的实现都能用，我们喜欢 OpenSSL。要生成这个证书：

```
server# openssl req -new -x509 -days 365 -nodes -out stunnel.pem -keyout
    stunnel.pem
Generating a 1024 bit RSA private key
.++++++
.....++++++
writing new private key to 'stunnel.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

www.TopSage.com

```
If you enter '.', the field will be left blank.
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Colorado
Locality Name (eg, city) [Newbury]:Boulder
Organization Name (eg, company) [My Company Ltd]:Booklab, Inc.
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
    server.example.com
Email Address []:
```

这条命令创建了一个自行签发、不用口令的证书。虽然不使用比较方便（也不必在每次 stunnel 重启的时候要人输入一个口令词），但也带来了一个安全风险。要用严格的权限来小心保护这个证书文件。

接下来，设定客户端和服务器端 stunnel 的配置。标准的配置文件是**/etc/stunnel/stunnel.conf**，但如果您想运行不止一条隧道，那么可以创建几个配置。

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = no

[telnets]
accept = 992
connect = 23
```

服务器端配置有两个要点。第一，注意 **chroot** 语句，它把 **stunnel** 进程限定在**/var/run/stunnel** 目录下。必要文件的路径需要按普通系统名字空间或者被 **chroot** 以后的名字空间来表示，具体取决于它们在哪里打开。此处的 **stunnel.pid** 文件实际上位于**/var/run/stunnel** 目录下。

`[telnetd]`端有两条语句：`accept` 告诉 **stunnel** 接受端口 992 上的连接，`connect` 把这些连接送到端口 23，即 Telnet 服务实际的端口。

客户端的配置也非常相似：

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = yes

[telnets]
accept = 23
connect = server.example.com:992
```

有两条指令反过来对应服务端的配置。`client =yes` 语句告诉程序发起 stunnel 连接而不是接受它们。本地的 stunnel 监听端口 23 上的连接，并连到服务器的 992 端口上。`connect` 指令中的主机名应该和创建证书时指定的输入项相匹配。

客户端和服务器端的 stunnel 都能不带命令行参数启动。如果您用 **netstat -an** 查看，应该看到服务器端 stunnel 等待 992 端口上的连接，而客户端 stunnel 在端口 23 上等待连接。

www.TopSage.com

大家网 TopSage.com

用户只要 telnet 到本地主机上，就能访问这个 stunnel 隧道：

```
client# telnet localhost 23
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Red Hat Enterprise Linux WS release 4 (Nahant Update 2)
Kernel 2.6.9-5.EL on an i686
login:
```

用户现在可以安全地登录，不必担心口令被窃取了。警惕的系统管理员会注意使用 TCP wrapper 来限制客户端上的连接只访问本地接口——目的不是让外界安全地 **telnet** 到服务器上！**stunnel** 是几种内置对 wrapper 功能支持的程序之一，它不要求使用 **tcpd** 来限制访问。浏览 www.stunnel.org 获得相关指导。

20.12 防火墙

除了保护单台机器之外，您还可以在网络的层面上实现安全预防措施。网络安全的基本工具是“防火墙”。防火墙主要分为 3 类：包过滤、服务代理和状态检查。

20.12.1 包过滤防火墙

包过滤防火墙根据包头中的信息，对可以通过 Internet 网关（或单位内部分隔域的内部网关）的流量类型做出限制。它很像您驱车在国际边境线上通过一个海关检查站。由您指定哪些目的地地址、端口号和协议类型是可接受的，网关只不过是丢掉了（在某些情况下则是记录下）不符合样板的数据包。

专用路由器，比如由 Cisco 制造的路由器就支持包过滤。它也可能由软件实现，这取决于您用作网关的机器及其配置。通常，包过滤防火墙会明显增强安全性，同时在性能或复杂性方面的损失很小。

Linux 包含了包过滤软件（参考 20.13 节的详细内容，了解更多的信息）。为了实现这一功能，还可以购买商业软件。这些软件包都有使用价值，而且他们能为家庭或者小型办公室提供一个相当安全的防火墙。不过，在您考虑用一个 Linux 系统来作为产品级的企业防火墙之前，应该参考一下本章开头给出的若干建议[9]。这是您真应该花钱买一台专用网络设备（比如 Cisco 的 PIX 防火墙）的地方。

20.12.2 如何过滤服务

大多数众所周知的服务都和**/etc/services** 文件或厂商给它的等价文件里某个网络端口相关联。提供这些服务的守护进程连接到适当的端口并等待来自远程站点的连接[10]。众所周知的服务端口中大多数是“特权”端口，也就是说它们的端口号在 1～1023 的范围中。这些端口只能由作为 root 运行的进程使用。1024 以上的端口号被称为非特权端口。

特定于服务的过滤机制则建立在这样的假定之上，即客户机（发起 TCP 或 UDP 会话的机器）将

[9] 我们假定您已经知道不考虑像 Windows 这样的操作系统来做防火墙的平台。“Windows”名字能唤起对安全的联想吗？太傻了吧，Windows 是给桌面用的。

[10] 在大多数情况下，实际上是 **xinetd** 在替它们等。更多信息请参见 28.3 节。

www.TopSage.com

使用非特权端口来联络服务器上的特权端口。例如，如果您希望只允许入站 SMTP 连接到地址为 192.108.20.200 的机器，请安装一个过滤器，允许目的地是该地址上端口 25 的 TCP 分组通过，并允许从该地址到任意地址的出站 TCP 分组通过[11]。安装这样一个过滤器的具体方法取决于您使用的路由器种类。

某些服务，比如 FTP，使得问题难上加难。FTP 协议在传送文件时实际上使用两个 TCP 连接：一个用于命令，另一个用于数据。客户机启动命令连接，服务器启动数据连接。因此，如果您想使用 FTP 从 Internet 检索文件，就必须允许到所有非特权 TCP 端口的入站访问，因为您不知道具体可能用哪个端口来构成一个传入数据连接。有关建立 **ftp** 服务程序的更多信息请参见 21.6 节。

这个麻烦很大程度上抵消了包过滤的意义，因为某些臭名昭著的不安全服务（例如，端口 6000 上的 X11）自然要和非特权端口绑定。这种配置还为机构内部好奇的用户提供了机会，让他们启动他们自己的服务（比如在非标准和非特权端口上的 **telnet** 服务程序），以便他们或他们的朋友可以从 Internet 上进行访问。

针对 FTP 的问题，一种常见的解决方案就是使用 SSH 的文件传输协议。这种协议现在是一种 Internet 草案，但得到了广泛的使用，而且很成熟。它经常作为 SSH 的一个子模块来用，SSH 提供了身份验证和加密功能。SFTP 和 FTP 不同，它使用一个端口既传命令又传数据，很好地解决了包过滤的矛盾问题。SFTP 的实现也有不少。我们已经用上了 OpenSSH 提供的命令行 SFTP 客户端。

如果您必须使用 FTP，那么一种合理的方法是只允许从这一台被隔离的主机 FTP 到外部世界。当用户需要执行内部网络禁止的其他网络操作时，他们也可以登录到这台 FTP 机器上。因为把所有用户账号复制到 FTP“服务器”上会破坏分隔管理的目标，所以您可能希望只在有人要求时才创建 FTP 账号。自然地，FTP 主机应该运行一个完整的安全性检查工具。

使用包过滤器最安全的方法，就是一开始的配置只允许 SMTP 或者 SSH 进入站点。然后当您发现有用的东西不能工作时再一点点儿地放宽过滤。

某些安全意识极强的站点使用双阶过滤机制。在这个方案中，第一个过滤器是到 Internet 的网关，第二个过滤器位于外部网关和本地网络的其余部分之间。这种想法是使外部网关相当开放，而让内部网关非常保守。如果将中间的机器与网络的其余部分分开管理，它就可以降低向 Internet 上提供各种服务时的风险。受到部分安全保护的网络通常叫做“非军事区”或者“DMZ”。

20.12.3 服务代理防火墙

服务代理将拦截去往和来自外部世界的连接，并建立到网络内部服务的新连接，它扮演的是两个世界之间的一种穿梭机或同伴角色。它很像驾驶汽车到国家的边境线，步行跨过边境，在边境的另一边租一辆消过毒的、刚清洗过的汽车继续行程。

服务代理防火墙的设计，使得它们比纯粹包过滤的灵活性要差得多（也慢得多）。代理必须有一个模块负责您希望允许通过防火墙的各种协议的解码和传送。在 20 世纪 90 年代早期这样做相当容易，因为只有几种常用的协议。如今，Internet 用户可能上网浏览一小时就使用了好几十种协议。结果，在使用 Internet 作为主要通信媒介的机构中，服务代理相对来说不太普遍。

20.12.4 状态检查防火墙

状态检查防火墙背后的理论是，只要您仔细监听和理解一个拥挤的机场上发生的所有会话（用所有的语言），就可以确定那一天是否有人打算引爆飞机。状态检查防火墙是为检查流过它们的流量，将

[11] 按照**/etc/services** 中的定义，端口 25 是 SMTP 端口。

www.TopSage.com

实际的网络活动与“应该”发生的情况相比较而设计的。例如，如果在 FTP 命令序列中交换的包指定了一个后来用于数据连接的端口，那么防火墙应该看到只在该端口上发生数据连接。远程站点连接到其他端口的企图将被推测成是伪造的，应该丢弃。

遗憾的是，现实通常是无情的。跟踪使用几百种协议的数千台主机的网络连接“状态”，比监听拥挤的机场上每种语言的每次谈话更不现实。有一天，当处理器和内存容量增加时，这也许最终会变得可行。

那么在厂商声称他们提供状态检查时，他们实际上卖的是什么呢？他们的产品不是监视的连接或协议的数目非常有限，就是搜索一组特殊的“坏”情形。不是说这样做有什么不对，显然，从任何可以检测出流量异常的技术中都可以获得好处。不过，在这种特殊情况下，重要的是记住厂商的宣传大多是促销手段。

20.12.5 防火墙保险吗

防火墙不应该是您防范入侵者的主要手段。它只适合作为一种安全措施的补充。防火墙的使用常常带来一种错误的安全感。如果它麻痹您，使您放松了其他安全措施，那么它对站点安全性的影响将是负面的。

单位内的每台主机都应该分别用像 **xinetd**、Nmap 和 samhain 这样的工具进行保护和定期监视。同样，您的整个用户群也需要进行基本安全知识的教育。否则，您只不过是建立了一个外强中干的架子而已。

理想情况下，本地用户应该能够连接到他们希望的任何 Internet 服务，而 Internet 上的机器应当只能够连接到一组受限制的本地服务。例如，您可能希望允许到本地存档服务器的 FTP 访问，并允许到邮件服务器的 SMTP（电子邮件）连接。

如果希望让 Internet 连接发挥最大的价值，我们建议您在决定如何设置网络时把重点放在方便性和可访问性上。不管怎么说，保证网络安全的是系统管理员的警惕性，而不是对防火墙硬件的幻想。

20.13 Linux 的防火墙功能: IP Tables

在正常情况下，我们并不建议把 Linux（或者 UNIX，或者 Windows）系统当作防火墙来用，因为功能全面的操作系统一般都不安全。专为路由器和包过滤设计的嵌入式设备（比如 Cisco 的 PIX 产品）才是最好的防火墙[12]。不过，对于一个家用站点，或者没有经费买合适硬件的站点来说，有 Linux 防火墙比根本没有还是要好，聊胜于无。

如果您正着手用一台 Linux 机器担当防火墙，请至少要确保它的安全配置和补丁方面已经是最新的。防火墙机器是落实本章所有推荐内容的好地方（从 20.12 节开始的内容在总体上讨论了包过滤防火墙。如果您不熟悉防火墙的基本概念，最好能在继续阅读下面的内容之前先读一下那一节）。

Linux 的 2.4 版内核引入了一种全新的包过滤引擎，称为 Netfilter。控制 Netfilter 的工具 **iptables** 是 Linux 2.2 版内核中比较老的命令 **ipchains** 的兄弟。**iptables** 把有次序的规则“链（chains）”应用到网络包上。链的集合就构成了“表（tables）”，用于处理特殊类型的流量。

例如，默认的 **iptables** 表名叫“filter（过滤器）”。这个表里的规则链都用于根据包来过滤网络流量。过滤器表包含 3 个默认链。内核处理的每个包都要经过 3 个链中适当的一个。FORWARD 链里的规则用于在一个网络接口收到的，而且需要转发到另一个网络接口的所有包。INPUT 和 OUTPUT 链

[12] 应该指出，许多面向消费的网络设备（如 Linksys 的路由器产品）的核心都使用 Linux 和 iptables。

www.TopSage.com

里的规则分别用于目的是本地主机，或者从本地主机发出的流量。这 3 个标准链通常能够满足两个网络接口之间防火墙的全部需要。如果必要，您还可以定义一个定制的配置，支持更复杂的记账或者路由场合。

除了 filter 表之外，**iptables** 还包含“nat”和“mangle”表。nat 表包含的规则链控制着 NAT（Network Address Translation，网络地址转换，这里的“nat”就是 iptables 表名，“NAT”是一般地址转换机制的叫法）。12.4.6 节讨论了 NAT，本节给出一个正在起作用的 nat 表的例子。在本节后面的内容里，我们会使用 nat 表的 PREROUTING 链来做反 IP 欺骗的包过滤。

mangle 表包含了链可以修改或者改变在 NAT 和包过滤之外的网络包的内容。虽然 mangle 表对于做包的特殊处理很方便，比如重新设定 IP 包的 ttl 值，但是在大多数工作环境中一般不会用到它。我们在这一节里只讨论 filter 和 nat 表，而把 mangle 表留给读者自己去探索。

构成一条链的每条规则都有一个“target（目标）”子句，它决定了如何处理匹配包。一旦某个包匹配了一个规则，那么它的结局就确定了，不再用其他规则来检查它。虽然 **iptables** 内部定义了许多 target，但是还可以指定另一条链作为规则的 target。

在 filter 表中的规则可以使用的 target 子句有 ACCEPT、DROP、REJECT、LOG、MIRROR、QUEUE、REDIRECT、RETURN 和 ULOG。当一条规则落在了 ACCEPT 上的时候，匹配这条规则的包就可以继续通过。DROP 和 REJECT 都会把它们的包丢弃。DROP 是默默地丢弃，而 REJECT 会返回一则 ICMP 错误消息。LOG 提供跟踪包匹配规则情况的简单办法，ULOG 提供了范围更广的日志记录信息。

REDIRECT 把包分流到一个代理，而不是直接让它们通过。您可能会用这个功能来迫使站点上的所有 Web 流量都通过一个 Web 高速缓存，比如 Squid。RETURN 终结用户定义的链，类似于子例程调用中的 return 语句。MIRROR 目标在发送包之前交换 IP 源和目的地址。最后，QUEUE 通过一个内核模块把包交给本地用户程序。

Linux 防火墙通常是由 **rc** 启动脚本中的一系列 **iptables** 命令来实现的。各个 **iptables** 命令通常采取下列形式之一：

```
iptables -F chain-name
iptables -P chain-name target
iptables -A chain-name -i interface -j target
```

第一种形式（**-F**）把以前所有的规则都从链中清除掉。第二种形式（**-P**）给链设置一条默认策略（也就是 target）。我们建议您使用 DROP 作为链默认的目标（target）。第三种形式（**-A**）把当前的说明追加到链上。除非您用**-t** 参数指定了表（table）名，否则您的命令就会被应用到 filter 表上。**-i** 参数将规则应用到有名字的接口上，**-j** 则标识出目标（target）。**iptables** 还接受许多别的子句，表 20.2 给出了其中的一部分。

表 20.2 iptables 过滤器的命令行标志

子 句	含义或者可能的取值
-p *proto*	匹配协议：**tcp**、**udp** 或者 **icmp**
-s *source-ip*	匹配主机或者网络源 IP 地址（CIDR 的记法也行）
-d *dest-ip*	匹配主机或者网络目的地址
--sport *port#*	匹配源端口（注意要用两个短划线）
--dport *port#*	匹配目的端口（注意要用两个短划线）
--icmp-type *type*	匹配 ICMP 类型代码（注意要有两个短划线）
!	否定一条子句
-t *table*	指定一条命令所应用的表（默认是 filter 表）

www.TopSage.com

下面我们剖析一个完整的例子。我们假定接口 ppp0 通往 Internet，接口 eth0 通往内部网络。ppp0 的 IP 地址是 128.138.101.4，eth0 的 IP 地址是 10.1.1.1，两个接口的子网掩码都是 255.255.255.0。这个例子使用无状态包过滤机制来保护 IP 地址为 10.1.1.2 的 Web 服务器，这是保护 Internet 服务器的标准方法。在这个例子的后面部分，我们将展示如何使用有状态的过滤机制来保护桌面用户。

在您使用 iptables 之前，必须启用 IP forwarding（IP 转发）功能，并且确保内核里已经加载了各个 **iptables** 模块。要了解启用 IP forwarding 功能的更多知识，参见 28.4 节或者 12.11.8 节。带有 **iptables** 的所有发行版本也都有能够完成启用和加载的启动脚本。

我们的第一组规则是对 filter 表进行初始化。首先，冲洗掉表中所有的链，然后将 INPUT 和 FORWARD 链的默认目标设为 DROP。和其他任何网络防火墙一样，最安全的策略就是丢弃您没有明确允许的任何包。

```
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

规则是按照它们出现在链中的顺序来进行匹配的，所以我们将用得最多的规则放在最前面[13]。FORWARD 链中的前 3 条规则让去往 10.1.1.2 上网络服务的连接通过防火墙。确切地说，我们允许 SSH（端口 22）、HTTP（端口 80）和 HTTPS（端口 443）能够连到我们的 Web 服务器。第一条规则允许来自可信网络的所有连接通过防火墙。

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j ACCEPT
```

我们允许流到防火墙主机（10.1.1.1）的唯一 TCP 流量是 SSH，它用于管理防火墙。下面列出的第二条规则允许环回（loopback）流量，它留在防火墙主机的本地。我们的系统管理员在他们不能 **ping** 到默认路由时会感到紧张，所以这里的第三条规则允许从内部 IP 地址来的 ICMP ECHO_REQUEST 包。

```
iptables -A INPUT -i eth0 -d 10.1.1.1 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i lo -d 127.0.0.1 -p ANY -j ACCEPT
iptables -A INPUT -i eth0 -d 10.1.1.1 -p icmp --icmp-type 8 -j ACCEPT
```

为了能让任何 TCP/IP 主机在 Internet 正常工作，必须允许某些类型的 ICMP 包通过防火墙。下面的 8 条规则就是让 ICMP 包既能送到防火墙主机，也能送到在它之后的网络的最小集合。

```
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 5 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 0 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 3 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 5 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 11 -j ACCEPT
```

我们接下来向 nat 表的 PREROUTING 链加入规则。虽然 nat 表的目的并不是做包过滤，但是它的 PREROUTING 链对于反 IP 欺骗的过滤来说特别有用。如果我们在 PREROUTING 链中加入 DROP 项，它们就不需要出现在 INPUT 和 FORWARD 链里了，因为 PREROUTING 链会应用到所有进入防火墙

[13] 不过，您必须要小心，不要为了性能在重新调整规则的顺序时改动了功能。

www.TopSage.com

大家网 TopSage.com

主机的包上。将控制项放到一个地方比重复放置它们的做法条理性要好得多。

```
iptables -t nat -A PREROUTING -i ppp0 -s 10.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 192.168.0.0/16 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 127.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 224.0.0.0/4 -j DROP
```

最后，我们用一条禁止所有没有得到明确许可的包的规则来结束 INPUT 和 FORWARD 链。虽然我们在前面已经用 **iptables -P** 命令实施过这个做法，但是 LOG 目标（target）能让我们看到谁正在从 Internet 上敲我们的大门。

```
iptables -A INPUT -i ppp0 -j LOG
iptables -A FORWARD -i ppp0 -j LOG
```

我们还可以设置 IP NAT 来伪装在内部网络里使用的私用地址空间。参见 12.12 节了解更多有关 NAT 的知识。

Netfilter 带给 Linux 防火墙最强大的功能之一就是有状态包过滤机制。针对连到 Internet 的客户机的防火墙不是允许特定的传入服务，而是允许根据客户机的请求而传入的响应。下面这条简单的有状态 FORWARD 链允许离开我们网络的所有流量通过，但只允许和我们的主机发起的连接有关的入流量通过。

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

要跟踪复杂的网络会话，比如 FTP 和 IRC，必须加载某些内核模块来启动 **iptables**。如果没有加载这些模块，**iptables** 就会简单地禁止那些连接。虽然有状态包过滤器能够提高站点的安全性，但是它们也增加了网络的复杂性。所以在防火墙上实现它之前，要确信您的确需要有状态功能。

调试 **iptables** 规则集最好的办法或许就是用 **iptables -L -v** 命令了。这些选项会告诉您链里面的每条规则匹配了多少个包。我们经常在想了解有关匹配包的更多信息时，增加带有 LOG 目标（target）的临时 **iptables** 规则。您可以经常通过使用像 **tcpdump** 这样的包探测器解决更棘手的问题。

20.14 VPN

过去几年中最有意思的发展之一就是 VPN（virtual private network，虚拟私用网络）的出现。这项技术之所以成为可能，主要归功于现在已经提高了的，可以在一个芯片（或者在用户的工作站）上提供的处理能力。VPN 最简单的形式是一条连接，这条连接让远程网络看上去仿佛就是直接连接的一样，哪怕它在物理上隔着千山万水，跨越了许多台路由器。为了提高安全性，这条连接不但以某种方式进行了验证（通常用像口令这样的一个“共享秘密”），而且还对端到端的流量做了加密处理。这样的编排通常称为“安全隧道”。

下面的这个例子就是一种体现了 VPN 方便之处的场合：假定有家公司在芝加哥、布德（译者注：作者工作的学校——科罗拉多大学的所在地）和迈阿密都有办事处。如果每个办事处都有一条到本地 ISP 的连接，那么这家公司就可以用 VPN 通过不可信的 Internet 把三地的办事处（而且在极大程度上是安全地）连接起来。这家公司可以租用连接三地办事处的专线来达到相同的效果，但是那样一来成本就太高了。

雇员在家远程办公的公司是另一个比较好的例子。VPN 可以让那些用户既享受到高速和廉价电缆

调制解调器（cable modem）服务的好处，又能让他们觉得仿佛是直接连接到了企业网。

因为这一功能的方便性和普及性，所以几乎所有的商家都能提供某种 VPN 的解决方案。您可以从路由器厂商那里把 VPN 当作是操作系统的一个插件，或者甚至是作为给您网络的一种专门 VPN 设备来购买它。根据预算和扩展性的需要不同，您可能会考虑市场上多种商业 VPN 解决方案的某一种。

如果没有预算，而又要找到一种能迅速解决问题的方案，那么 SSH 可以实现安全的隧道技术。在正常情况下，SSH 用于提供“每次一个端口”的连接，但是它也能像 12.13.9 节里给出的例子那样提供伪 VPN 功能，就是在一条 SSH 隧道上运行 PPP。

20.14.1 IPsec 隧道

如果您热衷于 IETF 的标准（或者热衷于省钱），又确实需要 VPN 解决方案，那么应该考虑一下 IPsec（Internet Protocol SECurity）。IPsec 最初是为 IPv6 开发的，但是它已经广泛为 IPv4 所实现。IPsec 是一种由 IETF 批准的端到端验证和加密系统。几乎所有重要的 VPN 厂商都提供至少一种兼容 IPsec 模式的产品。

IPsec 使用强加密技术，提供验证和加密服务。验证确保了包来自正确的发送方，在传送途中不会被篡改，而加密则防止未经授权对包的内容进行研究。

IPsec 目前的形式加密的是传输层的包头，它包括源和目的端口号。可惜这种方法和大多数防火墙的工作原理直接冲突。IETF 正在制订一项建议来取消这一功能。

2.5.47 及以后版本的 Linux 内核都自带一个 IPsec 实现，它和 2.4 版内核所采用的 FreeS/WAN 实现完全不同。因为 IPsec 是内核的一个组成部分，所以在我们所举例的所有 Linux 发行版本中都带有 IPsec。

注意，围绕 IPsec 隧道和 MTU 大小有一个难题。保证 IPsec 一次就把整个包加了密，而在隧道通过的路径上没有什么会对包进行分片是很重要的。为了做到这一点，可以降低隧道之前设备的 MTU（在现实世界里，1400 字节通常就行了）。参考 12.3.1 节，了解有关 MTU 大小的更多信息。

20.14.2 VPN 够用吗

可惜 VPN 也有不好的一面。虽然它们的确在两个端点之间不可信的网络中提供了一条（大多是）安全的隧道，但是它们往往没有解决端点本身的安全问题。例如，如果您在企业主干和 CEO（首席执行官）的家之间建立了一条 VPN 连接，您可能也会无意中为 CEO（首席执行官）15 岁的女儿建立了一条直接访问您的网络上任何地方的途径。她可能只会用她新得到的接入来和送货的伙计约会。

底线：您需要把 VPN 隧道连接当作外部连接来对待，只有在绝对必要，并且经过了认真考虑之后才赋予它们更多的权利。您可能想要考虑给您的网点安全策略中增加一个特殊的部分，这部分介绍对 VPN 连接应用什么样的规则。

20.15 强化安全的 Linux 发行版本

幸运的是，我们有了各种各样的创新，制造出“强化安全”的 Linux 版本，它们提供的安全功能要比主流发行版本的范围更广。“强化安全”版的 Linux 通常采取有特殊访问控制和审计功能的形式。如果您正打算在某种特制的网络设备产品上使用 Linux，这些功能可能就会显得格外有用了。不过，还不清楚它们是否确实把优势带给了大多数用户。它们仍然需要良好的维护和保养，一个良好的包过滤器，以及本章里讨论过的所有东西。或许它们只是增加了安慰。参考 20.10.6 节了解有关 SELinux

www.TopSage.com

的更多资料。

表 20.3 列出了比较知名的 Linux 加固项目，因此您可以看看它们能带给您什么。

表 20.3 加固的 Linux 发行版本

项目名称	Web 站点
Bastille Linux	www.bastille-linux.org
Engarde Linux	www.engardelinux.com
Openwall GNU/*/LINUX	www.openwall.com/Owl

20.16 如何对付站点攻击

对付攻击的关键很简单：不要恐慌。很有可能在您发现有人侵入的时候，大多数损害已经造成了。实际上，损害可能已经持续了几周或几个月。发现一个小时以前刚刚入侵的概率近似于零。

此时您应当做深呼吸，开始研究一个谨慎而周密的策略来对付入侵。您必须避免惊动入侵者，宣布被入侵或执行任何入侵者看起来不正常的活动都会惊动他们，他们可能已经观察您的站点运行好多个星期了。提示：此时执行系统备份通常是个好主意，这在入侵者看来（有希望！）是一种正常活动[14]。

此时也是提醒您自己的好时机，某些研究结果表明，60%的安全性事件与内部人员有关。在您确信自己掌握了所有事实之前和别人讨论此事时要小心。

下面是一个快速的九步计划，可能在您面临危急时会有所帮助。

步骤 1：不要恐慌。在大多数情况下，一个问题在发生数小时或数天之后才会被注意到。再过几个小时或几天不会影响结果。恐慌的反应和理性的反应之间则会不同。许多能恢复的情形因为在最初恐慌期间破坏了重要的日志、状态和跟踪信息而恶化了。

步骤 2：确定适度的反应。对安全事件反应过度不会让任何人受益。平静地进行即可。确定必须参与的人员和资源，让其他人在全部结束之后帮助进行事后剖析。

步骤 3：保留所有可用的跟踪信息。检查统计文件和日志。尝试确定入侵最初在何处发生。对整个系统进行备份。如果要从驱动器读备份，请确保在物理上给备份磁带做了写保护。

步骤 4：估计自己暴露的程度。确定有什么至关重要的信息（如果有的话）已经“离开”了公司，并设计适当的缓兵之计。确定未来的风险水平。

步骤 5：拔掉插头。如果有必要且合适的话，把受损害的机器从网络上断开。堵上已知的漏洞并止血。CERT 提供了分析一次入侵的步骤。从 www.cert.org/tech_tips/win-UNIX-system_compromise.html 可以找到这份文档。

步骤 6：制订一个恢复计划。和一个有创意的同事一起，在附近的白板上草拟一份恢复计划。这个过程离开键盘进行最为有效。把注意力集中在消除灾难和把损失降到最低上。避免评定过失或制造激动情绪。在计划里别忘记解决用户群可能经历的心理上的失落感。用户天生容易信任别人，明目张胆地破坏这种信任会让许多人感到难受。

步骤 7：沟通恢复计划。告诉用户和管理人员这次入侵的影响，未来潜在的问题，以及您预备的恢复策略。请尽量坦诚。安全事件是现代网络环境中生活的一部分。它们并不能反映您作为系统管理员的能力，或意味着任何别的值得难堪的事。坦白地承认您遇到了问题就赢得了 90%的胜利，只要您可以证明自己有补救的计划。

步骤 8：实现恢复计划。您比任何人都更熟悉自己的系统和网络。按照计划和本意去行事吧。和

[14] 如果在您的站点上系统备份不是“正常的”活动，那么您所面临的问题要比安全入侵事件严重得多。

www.TopSage.com

处于类似情况下的同事（最好是对您很了解的人）谈谈可以让自己不会脱离正轨。

步骤 9：向权威机关报告此次事件。如果事件牵涉到外部当事人，则应该向 CERT 报告该事件。请提供尽可能多的信息。

从 www.cert.org 可获得一份标准表单，这份表单有助于您理清思路。下面是您可能提供的一些比较有用的信息。

- 受害机器的名称、硬件类型和操作系统版本；
- 事件发生时已打过补丁的列表；
- 已知被损害的账号列表；
- 牵涉到的任何远程主机的名称和 IP 地址；
- 远程站点管理员的联络资料（如果您知道的话）；
- 相关的日志条目或审计信息。

如果您相信可能牵涉到一个在以前的文档中没有提到过的软件问题，则还应该向您的 Linux 发行商报告此次事件。

20.17 安全信息的来源

能大致跟上世界在安全方面的进展，保证系统安全也就成功了一半。如果站点被攻破，这不太可能是通过使用某种新技术做到的。更可能的是，在与安全性有关的新闻组和邮递列表中已经广泛地讨论过“您盔甲上的这个裂缝”。

20.17.1 CERT：卡内基梅隆大学的注册服务商标

作为对 1988 年 Internet 蠕虫猖獗行为的回应，DARPA（Defense Advanced Research Projects Agency，美国国防部高级研究计划署）成立了一个名为 CERT（Computer Emergency Response Team，计算机紧急响应组）的组织来充当计算机安全信息的情报交换场所。CERT 仍然是最为知名的安全信息联络点，虽然它似乎已经变得相当迟钝而且沾染了拖沓的官僚气。CERT 现在还坚持说 CERT 不代表任何意义，只是“卡内基梅隆大学的一个注册服务商标”。

在 2003 年，CERT 与美国国土安全部的 NCSD（National Cyber Security Division，美国国家网络安全局）合并了。合并后的机构改变了以前邮递列表的结构，这样做好坏参半。这个机构叫做 US-CERT，提供 4 个公告列表，其中最有用的是“Technical Cyber Security Alerts（技术方面的网络安全警报）”。订阅这 4 个邮递列表可以到 forms.us-cert.gov/maillists。

20.17.2 Security Focus.com 和 BugTraq 邮递列表

SecurityFocus.com 是一个与安全相关的新闻和信息的专门站点。新闻包括有关一般问题和特殊问题的最新文章。还有一个由有益的论文构成的内容广泛的技术库，技术库中的论文按照主题来排序。

SecurityFocus 的安全工具文档包括用于各种操作系统的软件，连同一起的还有吹捧的内容和用户的责骂。它是我们知道的最全面和最详细的工具来源。

BugTraq 列表是一个讨论有关安全漏洞及其修复的中等规模的论坛。如要订阅，可访问 www.securityfocus.com/archive。这个列表上的流量相当大，不过，信噪比相当糟糕。从这个 Web 网站还可以获得 BugTraq 漏洞报告的数据库。

www.TopSage.com

20.17.3 Crypto-Gram 时事通信

每月发布的 Crypto-Gram 时事通信是一个很有价值，而且有时又很富有情趣的信息资源，它涉及计算机安全和密码学。Crypto-Gram 时事通信是由 Bruce Schneier 提供的。从下面这个网站可以找到当前和以前的安全时事：

www.schneier.com/crypto-gram.html。

20.17.4 SANS：系统管理、网络和安全协会

SANS（系统管理、网络和安全）协会是一个专业组织，它举办与安全相关的会议和培训课程，并出版各种安全信息。他们的网站 www.sans.org 是一个有用的资源，内容介于 SecurityFocus 和 CERT 之间：既不像前者那么狂热，又不像后者那么难以消化。

SANS 提供几种电子邮件公告的周刊和月刊，您可以在他们的网站上登记加入。周刊 NewBites 内容丰富，而每月的总结则包含许多样板文件。两者都不是近期安全新闻的很好来源。

20.17.5 特定发行版本的安全资源

因为安全问题有可能造成很多不好的公共影响，所以各个厂商常常热心地帮助客户保持系统的安全性。大多数大型厂商都有一个官方的邮递列表，其中张贴有与安全相关的公告，以及与维护网站有关的许多安全问题。与安全相关的软件补丁通常是免费发布的，即使通常对软件支持收费的厂商也是如此。

在网上还有像 www.securityfocus.com 这样的安全门户网站，它包含针对厂商的信息和到厂商的最新官方意见的链接。



在 www.redhat.com/security 下可以找到 Red Hat 安全建议列表。在编写本书（英文版）的时候，Red Hat 没有发布正式的安全邮递列表。不过，在网上有各种各样的 Linux 安全资源，大多数信息都可以直接用于 Red Hat。



您可以在下面的地址找到 SUSE 的安全咨询信息：
www.novell.com/linux/security/securitysupport.html。



查看 www.debian.org，了解最新的 Debian 安全新闻。

有关 Cisco 产品的安全信息是以现场通知的形式来发布的，在 www.cisco.com/public/support/tac/fn_index.html 找到它的一份清单以及相关新闻。要订阅 Cisco 的安全邮递列表，可以发内容为“subscribe cust-security-announce”的邮件给 majordomo@cisco.com。

20.17.6 其他邮递列表和网站

上面列出的联系信息只是网络上可用的许多安全资源中的一小部分。但考虑到现在可用信息量很大，而资源又变化迅速，我们认为向您介绍一些基本资源会更有帮助。

Internet Security Systems 的 X-Force 网站是一个很好的起点，它维护着各种有用的 FAQ。其中有一份与安全相关的邮递列表的最新清单。厂商和安全补丁的 FAQ 则提供了各厂商的联络资料。

www.TopSage.com

大家网 TopSage.com

www.yahoo.com 列出了丰富的安全链接。在 www.wikipedia.org 下的“computer security”栏目中可以找到有关网络安全这一主题的另一个不错的链接资源。

Linux Journal（www.linuxjournal.com）有一个很棒的专栏叫“Paranoid Penguin（偏执的企鹅）”，这个专栏的内容覆盖了 Linux 安全的所有方面。该杂志偶尔也会包含有关安全专题的各种特色文章。

Linux Weekly News 是也很不错，其中会定期更新有关内核、安全、发行版本和其他专题的内容。在 lwn.net/security 可以找到 LWN 针对安全的部分。

20.18 推荐读物

BRYANT, WILLIAM. “Designing an Authentication System: a Dialogue in Four Scenes.” 1988. web.mit.edu/kerberos/www/dialogue.html

CERT COORDINATION CENTER. “Intruder Detection Checklist.” 1999.
www.cert.org/tech_tips/intruder_detection_checklist.html

CERT COORDINATION CENTER. “UNIX Configuration Guidelines.” 1997
www.cert.org/tech_tips/unix_configuration_guidelines.html

CHESWICK, WILLIAM R., AND STEVEN M. BELLOVIN. *Firewalls and Internet Security, 2nd Edition.* Reading, MA; Addison-Wesley. 2000.

CURTIN, MATT, AND MARCUS RANUM. “Internet Firewalls: Frequently Asked Questions.” 2004.
www.interhack.net/pubs/fwfaq

FARMER, DAN, AND WIETSE VENEMA. “Improving the Security of Your Site by Breaking Into it.” 1993. www.deter.com/unix/papers/improve_by_breakin.html

FARROW, RIK, AND RICHARD POWER. *Network Defense article series.* 1998-2004.
www.spirit.com/Network

FRASER, B., EDITOR. RFC2196: *Site Security Handbook.* 1997. www.rfc-editor.org.

BAUER, MICHAEL D. *Linux Server Security (2nd Edition)*. Sebastopol, CA: O’Reilly Media, 2005.

GARFINKEL, SIMSON, and GENE SPAFFORD. *Practical UNIX and Internet Security(3rd Edition).* Sebastopol: O’Reilly Media. 2003.

BARRETT, DANIEL J., RICHARD E. SILVERMAN, AND ROBERT G. BYRNES. *Linux Security Cookbook*. Sebastopol, CA: O’Reilly Media, 2003.

KERBY, FRED, ET AL. “SANS Intrusion Detection and Response FAQ.” SANS. 2003.

MANN, SCOTT, AND ELLEN L. MITCHELL. *Linux System Security: The Administrator’s Guide to Open Source Security Tools(2nd Edition).* Upper Saddle River, NJ: Prentice Hall PTR. 2002.

MORRIS, ROBERT, AND KEN THOMPSON. “Password Security: A Case History.” Communications of the ACM, 22 (11): 594-597, November 1979. Reprinted in *UNIX System Manager’s Manual, 4.3 Berkeley Software Distribution*. University of California, Berkeley. April 1986.

PICHNARCZYK, KARYN, STEVE WEEBER, AND RICHARD FEINGOLD. “UNIX Incident Guide: How to Detect an Intrusion.” Computer Incident Advisory Capability, U.S. Department of Energy. 1994. www.ciac.org/cgi-bin/index/documents

RITCHIE, DENNIS M. “On the Security of UNIX.” May 1975. Reprinted in *UNIX System Manager’s Manual, 4.3* Berkeley Software Distribution. University of California, Berkeley. April 1986.

SCHNEIER, BRUCE. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York, NY: Wiley, 1995.

www.TopSage.com

THOMPSON, KEN. "Reflections on Trusting Trust." in *ACM Turing Award Lectures: The First Twenty Years 1966-1985. Reading*, MA: ACM Press (Addison-Wesley). 1987.

SONNENREICH, WES, AND TOM YATES. *Building Linux and OpenBSD Firewalls*. New York, NY: J.W. Wiley. 2000.

这是一本小书：它易于阅读、有很好的例子、体现出了一种幽默感，而且自始至终都很不错。我们对这本书感到的唯一不足之处是，它反对使用 **sudo** 来进行 root 访问，称这种做法太难用，不值得那么费事。我们不太同意。

20.19 习题

E20.1 对比讨论用 Linux 口令的 SSH 身份验证和用口令词和密钥对的 SSH 验证。如果一种明显比另一种要安全，您自然应该采用更安全的验证方法吗？

E20.2 Samhain 能找出改动过的文件。

a）在您的机器上设置和使用 Samhain 需要什么？

b）Samhain 能有效地抵御 Internet 最近出现的什么弊病？

c）Samhain 能有助于抵御 Internet 最近出现的什么弊病？

d）如果能在物理上访问到系统，如何绕过 Samhain？

e）如果 Samhain 说**/bin/login** 被改动过了，但它的大小和修改日期似乎和以前一样，您会得出什么结论？如果 **sum** 程序给出的该程序新老版本的值是一样的，结论又如何？

★E20.3 SSH 隧道技术经常是把流量通过隧道送到远程的、您在上面没有管理员访问权限的计算机上的唯一方式。阅读 **ssh** 的手册页，给出一条命令，把流量用隧道从本地主机（localhost）的 113 端口送到 mail.remotenetwork.org 的 113 端口。隧道的转发点应该还是主机 mail.remotenetwork.org。

★E20.4 选择最近的一起安全事件，把它研究一番。找到有关此次事件最好的信息来源，寻找适用于您实验室里的系统的补丁或者变通办法。列出您的信息来源，以及为保护您的实验室建议采取的措施。

★★E20.5 在得到本地系统管理小组的同意之后，安装 John the Ripper，它是搜索薄弱登录口令的程序。

a）修改 John the Ripper 的源代码，只显示薄弱口令所关联的登录名，而不显示口令本身。

b)在本地实验机器的口令文件(您要有权访问**/etc/shadow**)上运行 John the Ripper，看看能找到多少可以破解出来的口令。

c）把您自己的口令设为一个字典里的词，然后只把您在**/etc/shadow** 文件中的配置项提供给 John the Ripper。**john** 要花多长时间找到它？

d）试试其他口令模式（大写字母、在字典里的单词后面加数字、单个字符的口令等），看看 **john** 到底有多聪明。

★★E20.6 在机房里，设置两台机器：一台目标主机，一台探测主机。

a）在探测主机上安装 **nmap** 和 Nessus。用这两种工具攻击目标主机。您怎样才能检测到目标上的攻击？

b）在目标主机上用 **iptables** 建立一个防火墙，防卫探测主机的攻击。您现在可以检测到攻击吗？如果能，怎样才能检测到？如果不能，为什么？

www.TopSage.com

c）抵御这种攻击还能用其他什么防卫手段？（要求有 root 权限。）

★★E20.7 安全小组最近发现在 **sendmail** 服务程序的许多当前和较早版本中都有一个大漏洞。找到有关漏洞的一个良好信息源，讨论这些问题以及解决它们的最好办法。

★★E20.8 setuid 程序有时虽然不好，但是有必要存在。不过，应该避免有 setuid 的 shell 脚本。为什么？

★★E20.9 使用 **tcpdump** 捕获主动和被动 FTP 会话的流量。支持一个匿名 FTP 服务器的需求会对站点的防火墙策略造成什么样的影响？防火墙的规则需要允许什么配置？（要求有 root 权限。）

★★E20.10 下面的 **iptable** 输出里的规则里允许什么，不允许什么？要增加安全性和保密性，容易添加哪些配置？（提示：OUTPUT 和 FORWARD 链可以多使用一些规则。）

```
Chain INPUT (policy ACCEPT)
target     prot  opt  source      destination
block      all   --   anywhere    anywhere

Chain FORWARD (policy ACCEPT)
target     prot  opt  source      destination
           all   --   anywhere    anywhere

Chain OUTPUT (policy ACCEPT)
target     prot  opt  source      destination

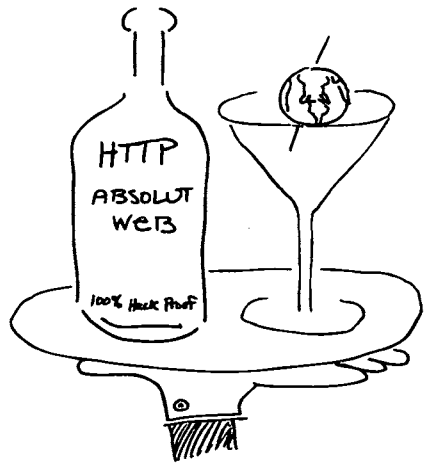
Chain block (1 references)
target     prot  opt  source          destination
ACCEPT     all   --   anywhere        anywhere     state RELATED,ESTABLISHED
ACCEPT     tcp   --   anywhere        anywhere     state NEW tcp dpt:www
ACCEPT     tcp   --   anywhere        anywhere     state NEW tcp dpt:ssh
ACCEPT     tcp   --   128.138.0.0/16  anywhere     state NEW tcp dpt:kerberos
ACCEPT     icmp  --   anywhere        anywhere
DROP       all   --   anywhere        anywhere
```

★★E20.11 检查一个本地防火墙的规则集。从策略的角度讨论您发现了什么。有什么明显的安全漏洞吗？（这一题需要负责本地站点安全的管理员予以配合。）

★★★★★E20.12 写一个工具，判断在您站点上的哪些网络接口处于混杂模式。在您的网络上定期运行这个工具，尝试迅速找出这样的入侵。这个工具造成多大的负载？您需要它在每台机器上都运行吗？还是可以从远程来运行它？您可以设计一个能暗中告诉您一个接口是否处于混杂模式的包吗？（要求有 root 权限。）

www.TopSage.com

第 21 章　Web 主机托管和 Internet 服务器



Web 技术的复杂性似乎每年都要增加一倍。还好的一点是，这项技术的绝大部分都在 Web 设计师和程序员掌握的领域内。而 Web 主机托管本身在过去十年里一直没怎么发生变化。

Web 服务器的软件产生出很多奇思妙想，结果这些服务器现在都变得相当安全和可靠——至少对于服务器配置正确，而且网站也没有捣乱的 Web 程序员的情况来说是这样。即便出现了“Web 2.0”、AJAX（Asynchronous JavaScript And XML，异步 JavaScript 和 XML）和动态 HTML 这样的技术，Web 服务器的核心软件仍然保持不变。

现如今，我们有五花八门的 Web 主机托管平台可供选择。微软的 Windows 作为一种 Web 主机托管平台占据了广大的市场。业界的新闻机构也发表了无数篇文章来质疑到底“哪种 Web 主机托管平台最好？”，通常站在拳台对角上的正是 Windows 和 Linux。虽然这种争夺颇有些广告战的味道（译者注，作者原文用了美国知名广告公司麦肯世界集团在 1974 年发表的著名广告词“Tastes great, less filling”），但是 Linux 已经因其低成本、快速、高可靠和灵活性，成为最流行的托管（hosting）平台。所谓 LAMP 平台（Linux、Apache、MySQL 和 PHP/Perl/Python）是当前 Web 服务器的主宰。

您可能希望不是在自己的站点上，就是在许多在同一地点提供外包服务的提供商之一那里“托管”许多围绕 Internet 开展的不同服务。在本章中，我们将介绍 2 种最常见的服务：Web 和 FTP。

21.1　Web 主机托管的基本知识

托管一个网站与提供任何其他网络服务在本质上并没有什么不同。WWW

www.TopSage.com

（World Wide Web，万维网）的基础是 HTTP（Hyper-Text Transfer Protocol，超文本传输协议），它是一种基于 TCP 的简单协议，用于传送包含文字、图片、声音、动画和视频在内的各种介质类型的文档。HTTP 的表现很像 Internet 上使用的其他客户机/服务器协议，比如 SMTP（用于电子邮件）和 FTP（用于文件传输）。

一台 Web 服务器只不过是配置为回答 HTTP 请求的系统。为了把一般的 Linux 系统变成 Web 主机托管平台，您需要安装一个在 TCP 端口 80（HTTP 的标准端口）上监听连接的守护进程，接受对文档的请求，并把文档传给发出请求的用户。

Web 浏览器，比如 Firefox、Opera 和 Internet Explorer，能代替用户联络远程 Web 服务器并发出请求。由此获得的文档可以包含指向（链接到）其他文档的超文本指针，指向的文档在用户最初联络的服务器上可能有也可能没有。因为 HTTP 协议标准定义得很好，所以运行在任何操作系统或体系结构上的客户机都可以连接到任何 HTTP 服务器。这种平台独立性加上 HTTP 可以把一个用户从一台服务器透明地传到另一台服务器的能力，帮助它取得了耀眼的成功。

不过，这类协议不只 HTTP 一种。现在已经定义了许多增强的协议，用于提供从加密到视频流的一切内容。这些增加的服务，即使它们在物理上由同一台服务器提供，但常常由单独的守护进程来管理。

21.1.1 统一资源定位符

统一资源定位符（uniform resource locator，URL）是指向 Internet 上某个对象或服务的一个指针。它描述如何通过 5 种基本组件访问一个对象：

- 协议或应用；
- 主机名；
- TCP/IP 端口（可选）；
- 目录（可选）；
- 文件名（可选）。

表 21.1 显示了 URL 中采用的一些协议。

表 21.1 URL 的协议

协　议	功　能	举　例
file	访问本地文件	file://etc/syslog.conf
ftp	通过 FTP 访问一个远程文件	ftp://ftp.admin.com/adduser.tar.gz
http	通过 HTTP 访问一个远程文件	http://admin.com/index.html
https	通过 HTTP/SSL 访问一个远程文件	https://admin.com/order.shtml
ldap	访问 LDAP 目录服务	ldap://ldap.bigfoot.com:389/cn=Herb
mailto	向指定地址发送电子邮件	mailto:linux@book.admin.com

21.1.2 HTTP 工作原理

HTTP 是让 WWW 真正工作的协议，而让许多人惊叹的是，它是一种极其基本的、无状态的客户机/服务器协议。在 HTTP 模式中，连接的发起人总是客户机（通常是一个浏览器）。客户机向服务器请求某个特定 URL 的“内容”。服务器不是用一组数据就是用某种类型的错误消息予以响应。客户机可以继续请求另一个对象。

因为 HTTP 是如此简单，您很容易就能使用 telnet 自制一个粗糙的 Web 浏览器。因为 HTTP 服务

www.TopSage.com

的标准端口是端口 80，只需直接 telnet 到选定的 Web 服务器的这个端口。连接上之后，就可以发出 HTTP 命令了。最常用的命令是 GET，它请求文档的内容。通常，GET/就是您想要的，因为它请求您所连接的任何服务器的根文档（通常即主页）。HTTP 区分大小写，所以一定要用大写键入命令。

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET /
<contents of your default file appear here>
Connection closed by foreign host.
```

更“完整”的 HTTP 请求包括 HTTP 协议的版本、要请求的主机（从基于名称的虚拟机检索一个文件时需要）和其他信息。接下来的响应包括信息报头以及响应数据。例如：

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.atrust.com

HTTP/1.1 200 OK
Date: Sun, 06 Aug 2006 18:25:03 GMT
Server: Apache/1.3.33 (Unix) PHP/4.4.0
Last-Modified: Sun, 06 Aug 2006 18:24:49 GMT
Content-Length: 7044
Content-Type: text/html

<contents of your default file appear here>
Connection closed by foreign host.
```

在本例中，我们告诉服务器我们要采用 HTTP 协议第 1 版进行通信，并且指出了要取得信息的虚拟主机的名字。服务器返回一个状态码（HTTP/1.1 200 OK），它认为的当前日期和时间，它所运行的服务器软件名称和版本号，被请求的文件上次修改的日期，被请求文件的长度，以及被请求文件内容的类型。HTTP 报头的信息和 HTTP 内容之间由一个空白行隔开。

21.1.3 CGI 脚本：即时生成内容

除了提供静态文档之外，HTTP 服务器可以向用户提供即时创建的内容。例如，如果您希望给拜访网站的用户提供当前的时间和温度，就可以让 HTTP 服务器执行一个脚本来获得此信息。这种让人吃惊的天生绝技通常是用 CGI（Common Gateway Interface，通用网关接口）实现的。

CGI 不是一种程序设计语言，而是一种允许 HTTP 服务器与其他程序交换信息的规范。编写 CGI 脚本最常用的语言有 Perl、Python 或者 PHP。但实际上，几乎任何可以执行实时 I/O 的程序设计语言都可以。只要想想那些无事可做的 COBOL 程序员现在可以把他们的技能重新应用到 Internet 就够了！

除了支持外部 CGI 脚本之外，许多 Web 服务器定义了一种插件结构，能让 Perl 和 PHP 这样的脚本解释器嵌入到 Web 服务器自身里面。这样的绑定极大地提高了性能，因为 Web 服务器不再需要派生出一个独立的进程去处理每个被请求的脚本。脚本开发人员基本上看不到这种体系结构。只要服务器看到一个文件是以一种特殊扩展名结尾的（比如.pl 或者.php），它就把这个文件的内容发给内嵌的解释器去执行。

CGI 脚本和插件在很大程度上与 Web 开发人员以及 Web 程序员有关。遗憾的是，CGI 脚本和系

www.TopSage.com

统管理员的工作在一个重要领域发生了冲突：安全性。因为 CGI 脚本和插件能够访问文件、网络连接和把数据从一个地方移动到另一个地方的其他方法，所以它们的执行有可能会影响运行 HTTP 服务程序的机器的安全性。最后，CGI 脚本或者插件赋予了世界上任何人在服务器上运行程序（脚本）的能力。因此，CGI 脚本以及插件所处理的文件需要和其他任何通过网络访问的程序一样安全才行。

21.1.4 负载平衡

很难预知一台服务器在单位时间内能够处理多少点击（对一个对象的请求，包括图像）或页面浏览（对 HTML 网页的请求）。服务器的容量取决于系统的硬件体系结构（包括子系统）、它所运行的操作系统、使用的任何系统调优措施的深度和广度，或许最为重要的一点是，站点采取的结构（它是纯静态的 HTML 网页，还是要进行数据库调用和数值计算的 HTML 网页？）。

只有对运行在实际硬件之上的实际站点进行直接评定和测试后，才能回答“多少次点击？”这个问题。有时，在相似的硬件上建立相似站点的人也许能给您提供模糊的猜测，这对做计划会有所帮助。您绝不应该相信商业系统提供商援引的数字。

还要记住，带宽也是一个关键因素。一台提供静态 HTML 文件和图像的机器提供的数据能轻而易举地占满一条 T3（45Mbit/s）链路。负载平衡既提高了性能又增加了冗余性。实现负载平衡的方法有多种：轮流（round robin）DNS、负载平衡硬件、以及基于软件的负载平衡工具。

应该指出，比单台服务器的点击量更好的指标是扩展性。一台 Web 服务器在用满其以太网卡之前，一般会变得非常消耗 CPU 或者 I/O。要保证您和您的 Web 设计团队有计划把流量负载很重的网站分散部署到多台服务器上。

轮流 DNS 是最简单和最原始的一种负载平衡方式。在这样的系统上，给一个主机名分配了多个 IP 地址。当名字服务器收到查询该 Web 网站 IP 地址的请求时，作为响应，客户机会收到其中一个 IP 地址。地址按一种重复的“循环”方式，一个接一个轮流回应给客户机。有关轮流 DNS 的更多信息可参考 15.5.2 节的内容。

轮流 DNS 也有问题，如果服务器宕机了，必须更新 DNS 数据，把服务器从循环响应中删除。远程缓存的 DNS 数据将导致这一操作不稳定可靠。如果有备份服务器，那么把坏了的服务器的 IP 地址重新分给备份服务器往往要容易得多。采用负载平衡硬件是另一种相当简便可行的方法，但这要求有些富余的资金。商业的第三方负载平衡硬件有 F5 Networks 公司的 Big-IP Controller，Nortel 公司的 Web 交换产品，以及 Cisco 公司的 Content Services Switches（内容服务交换机）。这些产品按照各种可配的参数把传入站点的请求量分散到多台设备上，并且在分发请求的时候能够考虑到各服务器的响应时间。

基于软件的负载平衡工具不要求用专门的硬件，它们可以在一台 Linux 服务器上运行。这类软件有商业的也有开放源代码的。开放源代码的一类软件包括 Linux Virtual Server、Ultra Monkey 和 Apache 的模块 mod_backhand。商业软件的例子有 Zeus 公司销售的那些产品。

您或许会感到好奇，想知道像 Google 这样的大型网站是怎样处理负载平衡问题的。它们的系统把定制的负载平衡 DNS 服务器和负载平衡硬件组合起来一起使用。

要记住一点，现在大多数网站的内容都是动态生成的。这样的体系结构给数据库服务器带来了很重的负载。如果必要的话，要咨询数据库管理员来决定在多台数据库服务器之间分散负载的最佳方案。

21.2 HTTP 服务程序的安装

安装和维护 Web 服务程序非常容易。Web 服务在复杂性和管理困难度上远小于电子邮件和 DNS。

www.TopSage.com

21.2.1 选择服务器软件

有几种优秀的 HTTP 服务程序，但是您可能最愿意从 Apache 服务器起步，Apache 因其灵活性和高性能在业界非常知名。到 2006 年 9 月，Internet 上 63%的 Web 服务器在运行 Apache。剩下的服务器中，微软占了大部分，即 30%。在过去 5 年的时间里，Apache 和微软在这一市场中的份额比例保持得相当稳定。从 news.netcraft.com/archives/web_server_survey.html 可以了解过去以来详细的市场份额统计数据。

您可以在 www.serverwatch.com/stypes/index.php（选择“Web Servers”）这个网站上找到所有当前可用的 HTTP 服务程序的比较。下面是您在决策时可能要考虑的一些因素。

- 健壮性；
- 性能；
- 更新和修正错误的及时性；
- 是否能获得源代码；
- 商业支持或者社群支持力度；
- 费用；
- 访问控制和安全性；
- 充当代理的能力；
- 处理加密的能力。

Apache HTTP 服务程序“是个免费的好网站软件”，完整的源代码可从 Apache Group 的网站 www.apache.org 下载。不喜欢冒险的人可能愿意安装随 Linux 发行版本带的二进制的 Apache 软件包（但是很可能它已经安装到您的系统上了，看一下有没有**/etc/apache2**）。

21.2.2 安装 Apache

如果您决定自己下载源代码并编译它，就需要执行软件发布中带的 **configure** 脚本。这个脚本自动检测您所用的系统类型，并构造合适的 makefile 文件。您需要用**--prefix** 选项来指定 Apache 服务器应该呆在系统目录树里的什么地方。如果没有指定一个目录前缀，那么服务程序就会默认安装在**/usr/local/apache2**。例如：

```
$ ./configure --prefix=/etc/httpd/
```

您可以使用 **configure --help** 看到所有选项的完整清单，其中大多由**--enable-***module* 和**--disable-***module* 这样的选项构成，它们可以把 Apache 的一些功能包括在服务程序中或从服务程序中删除。

您还可以通过指定**--enable-***module***=shared** 选项（或者使用**--enabled-mods-shared=all** 让所有的模块都成为共享模块）把模块编译成动态共享对象。这样一来，以后就可以指定要包括还是不包括哪些模块，而在运行时只加载 **httpd** 配置中的模块。对于二进制的 Apache 软件包来说，默认就采取了这样的配置——所有的模块都编译成共享对象，在 Apache 启动的时候动态加载。使用共享库的唯一缺点是启动时间稍微有点儿长，性能上略微有所下降（一般降低 5%）。对于大多数网站来说，因为随时能够加上新模块，不必重新编译就能关闭现有模块而获得的好处远远超过了性能上的损失。

标准模块的完整列表可参见 httpd.apache.org/docs-2.0/mod。虽然默认设置的一组模块比较合理，但您可能还想启用表 21.2 中列出的其他模块。

www.TopSage.com

表 21.2 默认不启用的 Apache 模块

模 块	功 能
auth_dbm	使用 DBM 数据库来管理 usr/group 访问（如果您要求逐个用户以口令方式访问 Web 网站的内容则推荐采用）
rewrite	使用正则表达式重写 URL
expires	让您给文档附加作废日期
proxy	用 Apache 作为代理服务器（后面会做更多说明）
ssl	启动对 SSL（Secure Socket Layer，安全套接口层）的支持（用于 HTTPS）

同样，您可能希望禁用表 21.3 中列出的模块。为安全和性能，把已知不会用到的模块禁用是个好主意。

表 21.3 我们建议删除的 Apache 模块

模 块	功 能
asis	允许指定的文件类型不带 HTTP 报头发送
autoindex	给没有默认 HTML 文件的目录建索引
env	让您为 CGI 脚本设置特殊的环境变量
include	允许服务器端包含（SSI），一种即时创建内容的方式
userdir	允许用户有自己的 HTML 目录

在 **configure** 执行完以后，运行 **make**，然后再运行 **make install**，进行实际的编译工作，并安装适合的文件。

21.2.3 配置 Apache

一旦您已经安装好了服务程序，那么就需要为您的环境来对它进行配置了。所有配置文件都保存在子目录 **conf** 中（例如**/usr/local/apache2/conf**）。您需要检查并自定制 **httpd.conf** 这个文件，它分成 3 节。

第一节配置全局设定，比如服务器池、HTTP 服务器监听查询的 TCP 端口（通常是 80 端口，但您也可以选择其他端口——而且可以在一台机器的不同端口上运行多个 HTTP 服务器），以及动态模块加载的设定。

第二节配置“默认”服务器，即根据 `VirtualHost` 定义（参见 21.3.3 节）无法应答的所有请求都由这个服务器来受理。在这一节里配置的参数包括服务器以哪个用户和用户组的身份运行（要以非 root 的身份!），还有非常重要的 `DocumentRoot` 定义，它规定了对外提供的文档所在目录树的根。这一节还解决诸如“特殊”URL（像那些以*~user* 语法表示用户主目录的 URL）的处理这样的问题。

在配置文件的第二节里还要管理安全方面的问题。可以逐个文件（`<File>`指令）或者逐个目录（`<Directory>`）地来控制访问。这些权限设定避免了通过 **httpd** 访问到敏感文件。您应该至少指定两种访问控制：一种覆盖整个文件系统，另一种适用于主文档目录。虽然我们建议您删掉 `AllowSymLinks` 这个选项，不让 **httpd** 在目录树中能够沿着符号链接访问，但 Apache 默认的设定也够安全了（我们不想让人意外地创建一个**/etc** 的符号链接，不是吗？）。要获得更多有关 Apache 安全的小窍门，可参考 httpd.apache.org/docs-2.0/misc/security_tips.html。

配置文件的第三节也是最后一节设置虚拟主机。我们将在 21.3.3 节详细再谈。

在修改好配置之后，运行 **httpd -t** 检查配置文件的语法。如果 Apache 报告说“Syntax OK”，那么就算配好了。如果没有，则要检查 **httpd.conf** 文件里有没有敲错的地方。

www.TopSage.com

21.2.4 运行 Apache

您可以手工或从机器的启动脚本启动 **httpd**。后一种方法更好，因为这种配置将确保不论机器何时重新引导，Web 服务程序都会重新启动。为了手工启动该服务程序，请键入下面这样的命令：

```
$ /usr/local/apache2/bin/apachectl start
```

如果希望 **httpd** 在系统引导时自动启动，请在 **rc** 目录下建一个链接，指向**/etc/init.d/httpd** 文件（它是作为 **httpd** 软件包的一部分安装的）。最好在引导序列的后面部分，即管理路由选择、时间同步这样功能的守护进程之后启动 **httpd**。参考第 2 章了解有关 **rc** 脚本的更多信息。

21.2.5 分析日志文件

当您的 Web 网站作为生产系统运行的时候，您可能想要收集有关网站使用情况的统计数据，比如每个网页被访问的次数、每天平均的访问量、访问失败的比例、以及数据传输量等。要确保您正在使用日志的“combined（组合）”格式（`CustomLog` 指令的后面是 `combined` 这个词而不是 `common`）。这种日志格式包含了每次访问请求从哪里来（URL 从哪个网页链过来的）和用户代理（客户机的浏览器和操作系统）的信息。

访问和出错日志位于 Apache 的 **logs** 目录下。日志文件可以直接供人阅读，但是它们包含的信息过多，您确实需要一个单独的分析程序从中提取有用的数据。毫不夸张地说，这样的日志分析器，包括商业的和免费的，加起来有数百种之多。

值得考虑的两种免费日志分析器有 Analog 和 and AWStats。这两种软件都提供了相当基础的信息。如果您想要让报告更丰富多彩，那可能要买一种商业软件。从 www.practicalapplications.net/kb/loganalysis.html 可以找到一份有帮助的软件清单。

21.2.6 高性能主机托管的静态内容优化

主机托管界在近几年来认识到，创建高性能主机托管平台最简单的方法之一就是针对托管的静态内容优化服务器。Linux 是唯一能在这个领域有所作为的系统，它需要利用 TUX Web 服务器。

TUX 是一种基于内核的 Web 服务器，它配合传统的 Web 服务器，比如 **Apache** 一同运行。只要有可能，TUX 服务器就会在不离开内核空间的情况下提供静态页面服务，这和 **rpc.nfsd** 提供文件服务的方式比较类似。这种体系结构不需要把数据在内核空间和用户空间之间进行拷贝，从而把上下文切换的次数降至最低。虽然并不推荐初学者使用 TUX，但是对于需要快速提供静态内容服务的网站来说，它是一种非常好的选择。

虽然 TUX 是由 Red Hat 开发的（从 www.redhat.com 可以获得它），但是它已经按照 GPL 来发布，其他 Linux 发行版本上也能使用它。不过，配置 TUX 有一定的挑战性。了解详情请参考：

http://www.redhat.com/docs/manuals/tux。

21.3 虚拟接口

早先，一台机器一般只充当一个网站的服务器。随着 Web 的日益流行，每个人都希望有自己的网站，一夜之间，数千家公司变成了 Web 主机托管的提供商。

www.TopSage.com

大家网 TopSage.com

提供商们很快认识到，如果他们能够把多个站点放在一台服务器上，在经济上就可以获得显著的规模效应。这一技巧能够让相同的硬件透明地向 www.acme.com、www. toadranch.com、以及许多其他站点提供服务。虚拟接口应这种业务的需求而诞生了。

虚拟接口使一个守护进程不仅能够根据目的端口号（例如，用于 HTTP 的 80 端口），还能根据连接的目的 IP 地址来识别连接。如今，虚拟接口得到了广泛使用，而且经证明可以用于 Web 主机托管之外的其他应用。

其思想很简单：一台机器在网络上对比它的物理网络接口更多的 IP 地址作出响应。每个由此得到的“虚拟”网络接口可以被关联到用户在 Internet 上希望连接到的一个对应域名。这种功能使得一台机器能够切实地向几百个网站提供服务。

21.3.1 使用基于名字的虚拟主机

HTTP 1.1 协议定义了一种类似于虚拟接口的功能（正式名字叫做“基于名字的虚拟主机”），它消除了需要给 Web 服务程序分配唯一的 IP 地址或在操作系统层面上配置特殊接口的要求。这种方法节省了 IP 地址，对某些站点，特别是一台服务器容纳了数百或数千个主页（比如大学）的站点非常有用。

不过，这一方案对商业站点不是很实用，它降低了可扩展性（必须修改站点的 IP 地址才能把它迁移到另一台服务器上），而且还可能对安全性造成负面影响（如果防火墙是根据 IP 地址过滤对站点的访问的话）。此外，基于名字的虚拟主机不能使用 SSL。看起来，真正的虚拟接口尚需等一段时间才会出现。

21.3.2 配置虚拟接口

建立一个虚拟接口包括两个步骤。首先，必须在 TCP/IP 层次上创建虚拟接口。其次，必须告诉 Apache 服务程序您已经安装的虚拟接口。我们将从 21.3.3 节开始讨论第二个步骤。

Linux 虚拟接口按 *interface:instance* 的方式命名。例如，如果您的以太网接口是 eth0，那么与之相关联的虚拟接口就应该是 eth0:0，eth0:1，依此类推。所有的接口都用 **ifconfig** 命令来配置。例如，命令

```
# ifconfig eth0:0 128.138.243.150 netmask 255.255.255.192 up
```

配置接口 eth0:0，给它分配一个网络 128.138.243.128/26 上的地址。



为了让 Red Hat 和 Fedora 上分配的虚拟地址永久有效，您可以在**/etc/sysconfig/network-scripts** 目录下为每个虚拟接口创建一个文件。例如，对应上面那条 **ifconfig** 命令的 **ifcfg-eth0:0** 文件内容是：

```
DEVICE=eth0:0
IPADDR=128.138.243.150
NETMASK=255.255.255.192
NETWORK=128.138.243.128
BROADCAST=128.138.243.191
ONBOOT=yes
```



Debian 和 Ubuntu 的方法和 Red Hat 的方法类似，但是接口定义必须出现在文件 **/etc/network/interfaces** 中。和我们上面的例子中 eth0:0 接口相对应的配置项应为：

```
iface eth0:0 inet static
    address 128.138.243.150
    netmask 255.255.255.192
    broadcast 128.138.243.191
```

www.TopSage.com



在 SUSE 系统上，您要么可以用 YaST 创建虚拟接口，要么可以手工编辑接口文件来创建它们。

在 SUSE 环境下，一个接口的所有 IP 地址都在一个文件里配置。手工配置这些文件的话，要在**/etc/sysconfig/network** 目录下找名字以 **ifcfg-*ifname*** 开头的文件。实际接口的文件名带有麻烦的 6 字节 MAC 地址，那些文件就是您要找的文件。

例如，要定义两个虚拟接口，其中一个配置文件要包括下面这些配置项：

```
IPADDR_0=128.138.243.149
NETMASK_0=255.255.255.192
LABEL_0=0
IPADDR_1=128.138.243.150
NETMASK_1=255.255.255.192
LABEL_1=1
STARTMODE="onboot"
NETWORK=128.138.243.128
```

IPADDR 和 NETMASK 后面的后缀（这里是_0 和_1）不一定是数字，但为了保持一致，这是一个合理的约定。

21.3.3 告诉 Apache 有关虚拟接口的信息

除了创建虚拟接口之外，您还需要告诉 Apache，当一个客户机试图连接到各个接口（IP 地址）的时候，应该给它提供什么文档。用 **httpd.conf** 文件里的 `VirtualHost` 子句就能完成配置。每个已经配置好的虚拟接口都要对应一条 `VirtualHost` 子句。下面是一个例子：

```
<VirtualHost 128.138.243.150>
    ServerName www.company.com
    ServerAdmin webmaster@www.company.com
    DocumentRoot /var/www/htdocs/company
    ErrorLog logs/www.company.com-error_log
    CustomLog logs/www.company.com-access_log combined
    ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

这个例子把目录**/var/www/htdocs/company** 下的文档提供给连接到虚拟主机 128.138.243.150 的任何客户机。几乎所有的 Apache 指令都能引入一个 `VirtualHost` 子句，规定特定于虚拟主机的设定。相关的目录路径（包括 `DocumentRoot`、`ErrorLog` 和 `CustomLog` 指令的那些目录路径）都以 `ServerRoot` 为参照来解释。

有了基于名字的虚拟主机，多个 DNS 域名就可以指向同一个 IP 地址。Apache 的配置也类似，但您要指定 Apache 应该在哪个主 IP 地址上监听发来的带有虚拟主机名的请求，并且在 `VirtualHost` 子句中排除这个 IP 地址：

```
<VirtualHost *>
    ServerName www.company.com
    ServerAdmin webmaster@www.company.com
    DocumentRoot /var/www/htdocs/company
    ErrorLog logs/www.company.com-error_log
    CustomLog logs/www.company.com-access_log combined
    ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

在上述配置中，Apache 检查 HTTP 报头得知请求访问的网站。服务器在其主 IP 地址

www.TopSage.com

128.138.243.150 上监听对 www.company.com 的请求。

21.4 SSL

SSL（Secure Socket Layer，安全套接口层）协议负责 Web 网站和客户端浏览器之间的通信安全。以 https://开头的 URL 就用到了这项技术。SSL 利用加密来防止窃听、篡改和假冒消息。

浏览器和服务器使用基于证书的身份验证机制建立通信，之后它们切换到一种速度更快的基于密码的加密方式，以此保护实际的通信内容。

SSL 在 HTTP 应用协议之下作为一个独立的协议层运行。SSL 只保证连接的安全性，它本身并不参与 HTTP 事务。正是由于有了这种很干净的体系结构，SSL 不仅能够保护 HTTP 的安全，而且还能保护 SMTP、NNTP 以及 FTP 等协议的安全。

在 SSL 的“早期”使用中，大多数对称加密密钥是相对较为薄弱的 40 位密钥，因为美国政府限制密码技术的出口。经过多年的争论和诉讼之后，政府放宽了有些方面的出口限制，允许 SSL 的实现使用 128 位的对称加密密钥。参见 30.12.1 节了解围绕加密产生的法律问题。

21.4.1 产生签发证书的请求

使用 SSL[1]的 Web 网站拥有者必须产生一个 CSR（Certificate Signing Request，签发证书请求），CSR 是一个数字文件，其中包含一个公钥和一个公司名称。之后，“证书”必须由一个称为 CA（Certificate Authority，证书授权机构）的可信来源签发。CA 返回的签发证书包含网站的公钥和公司名称，还有 CA 的背书。

Web 浏览器内置有 CA 列表，它们接受由这些 CA 签发的证书。浏览器如果认可给您的网站签发证书的 CA，那么它就会核对您的证书，取得您的公钥，然后用这个公钥发送只有您的网站能解密的消息。虽然您实际上也可以自己签发证书，但如果证书不是从得到认可的 CA 取得的，那么大多数浏览器都会告诉用户证书可疑。在商业环境下，这样的表现显然有问题。但是如果您想给内部用户和测试工作建立自己的 CA，那么可以参考 httpd.apache.org/docs/2.0/ssl/ssl_faq.html#aboutcerts。

您可以从若干个 CA 中的一个签发证书。在 google 里输入“SSL certificate”就能查到。CA 之间唯一真正的区别在于它们核实您的身份所做的工作量、它们提供的保险度、以及外部支持它们的浏览器的数量（大多数 CA 都得到了大部分浏览器的支持）。

创建一个证书发到 CA 相当简单。必须安装 OpenSSL，在大多数 Linux 发行版本上默认都安装了这个软件。下面是具体步骤。首先为 Apache 服务器创建一个 1024 位的 RSA 私钥：

```
$ openssl genrsa -des3 -out server.key 1024
```

这里会提示输入并确认加密服务器密钥的口令字。把 **server.key** 这个文件备份到一个安全的地方（只有 root 可读），并且要记住您输入的加密口令字。好奇的话，可以用下面的命令看看密钥的详细数字内容：

```
$ openssl rsa -noout -text -in server.key
```

接下来，创建一个 CSR，把刚才生成的服务器密钥加进这个 CSR：

[1] TLS（Transport Layer Security，传输层安全）是 SSL 的后继，在现代所有的浏览器上都已实现。不过，Web 界仍然把整个协议/概念称为 SSL。

www.TopSage.com

```
$ openssl req -new -key server.key -out server.csr
```

在提示您输入“common name（常用名）”的时候，输入服务器的完整域名。例如，如果您的网站的 URL 是 https://www.company.com，就输入“www.company.com”作为常用名。注意，您需要给每个主机名一个单独的证书——甚至“company.com”和“www.company.com”都要区分。公司一般只注册一个常用名，它们把任何基于 SSL 的连接都指向这个主机名。

您可以用下面的命令产查看生成好的 CRS 的详细内容：

```
$ openssl req -noout -text -in server.csr
```

现在您可以把 **server.csr** 这个文件发给您选出的 CA，由这个 CA 签发证书。没必要在本地保留这个文件的副本。CA 返回的签发过的 CSR 应该以.**crt** 为扩展名。把这个签发过的证书放在和私钥相同的安全的地方。

21.4.2 配置 Apache 使用 SSL

HTTP 请求发到 80 端口，HTTPS 请求则使用 443 端口。HTTPS 和 HTTP 流量都由 Apache 进程提供服务。不过 SSL 不能和基于名字的虚拟主机配合使用，每个虚拟主机都必须有一个专门的 IP 地址（这个限制源自于 SSL 的设计）。

要配置 Apache 使用 SSL，首先要在 httpd.conf 文件里启用 SSL 模块，做法是找到或者增加这么一行内容：

```
LoadModule ssl_module    libexec/mod_ssl.so
```

然后给 SSL 端口加上一个 VirtualHost 指令：

```
<VirtualHost 128.138.243.150:443>
    ServerName www.company.com
    ServerAdmin webmaster@www.company.com
    DocumentRoot /var/www/htdocs/company
    ErrorLog logs/www.company.com-ssl-error_log
    CustomLog logs/www.company.com-ssl-access_log combined
    ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
    SSLEngine on
    SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
    SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key
</VirtualHost>
```

注意 IP 地址后的:443 和告诉 Apache 到哪儿找到您的私钥和签发证书的 SSL 指令。

当您重新启动 Apache 的时候，它会要求输入用于 **server.key** 文件的口令词。因为这里要人机交互，所以系统引导的时候，**httpd** 再不能自动启动了。如果愿意，可以去掉对私钥的加密，避开输入口令的步骤：

```
$ cp server.key server.key.orig
$ openssl rsa -in server.key.orig -out server.key
$ chmod 400 server.key server.key.orig
```

当然，任何人只要获得了未加密的密钥，都能假冒您的网站。

要了解更多有关 SSL 的信息，可参考以下资源：

httpd.apache.org/docs-2.0/ssl/ssl_faq.html

httpd.apache.org/docs/2.0/mod/mod_ssl.html

www.TopSage.com

21.5 缓存和代理服务程序

Internet 和它上面的信息正在迅速增长。因此，支持它所需的带宽和计算资源也在迅速增长。这种状态该怎样维持下去呢？

对付这种增长唯一的方法就是使用复制。无论是在国家、地区还是站点的层次上，随着 Internet 的增长，Internet 的内容需要从更近的来源更容易地获得。每天数百万次地把同一流行网页通过一条非常昂贵的链路从澳大利亚传到北美，这是毫无意义的。应该有一种方法，只要它通过链路发送一次，就把这些信息保存下来。幸运的是，有这样的方法。

21.5.1 Squid 缓存和代理服务器

一种答案是免费的 Squid Internet Object Cache[2]。这个软件包既是一个高速缓存服务程序，也是一个代理服务程序，支持包括 HTTP、FTP 和 SSL 在内的好几种协议。

下面介绍它是如何工作的。客户端的 Web 浏览器连接到 Squid 服务程序，请求获得一个 Internet 上的对象。然后 Squid 服务程序代表客户提出请求（或者从它的高速缓存中提供对象，如同在下面的段落中讨论的那样）并把结果返回给客户机。这种类型的代理服务程序常常被用来增加安全性或过滤内容。

在基于代理的系统上，只有一台机器需要通过单位的防火墙直接访问 Internet。在像 K-12（译者注：意思是指从幼儿园[kindergarten]到 12 年级）学校这样的单位里，代理服务程序还可以过滤内容，这样，不合适的资料就不会落入错误的人手中。如今有许多商业的和免费的代理服务程序（一些以 Squid 为基础，另一些则不是）。这些系统有些基于纯软件（像 Squid），有些则采用了硬件设备（例如，BlueCoat）的形式。在 www.web-caching.com/proxy-caches.html 可以找到更全的代理服务器技术清单。

代理服务不错，但真正值得令人兴奋的是 Squid 的高速缓存功能。Squid 不只是高速缓存来自本地用户请求的信息，还能让 Squid 服务器组成一种层次结构[3]。Squid 服务器集群使用 ICP（Internet Cache Protocol，Internet 缓存协议）来沟通其高速缓存中的信息。

这项功能使管理员能够建立一个系统，在这个系统中本地用户联络本地的高速缓存服务程序以获取来自 Internet 的内容。如果本地的另一个用户已经请求过相同的内容，那么就能以 LAN 的速度（通常为 100Mbit/s 甚至更高）返回一个副本。如果本地的 Squid 服务器里没有这个内容，也许它会联系地区的缓存服务器。和本地的情况一样，如果该地区中有任何人请求过该对象，就会立即提供它。如果没有，也许可以联系用于本国或本大洲的缓存服务器，依此类推。用户感觉到性能有所改善，于是会很高兴。

对许多人来说，Squid 提供了经济上的好处。因为用户对 Web 的探索有趋同性，所以在适度规模的网点上，会出现相当多重复的对外 Web 请求。一项研究结果表明，运行高速缓存服务器可以减少多达 40%的对外带宽需求。

为了有效地利用 Squid，您可能想要强迫您的用户使用高速缓存。方法有，通过 Active Directory 配置一个代理（在基于 Windows 的环境中），或者使用 WCCP（Web Cache Communication Protocol，Web 高速缓存通信协议）配置路由器把所有基于 Web 的流量重定向到 Squid 的高速缓存上。

[2] 为什么叫做“Squid”？根据 FAQ 的解释，“所有的好名字都用完了。”

[3] 遗憾的是，有些站点把它们所有的网页都标为不可缓存，这就无法让 Squid 发挥它的神奇作用。类似地，Squid 也不能缓存动态生成的网页。

www.TopSage.com

TopSage.com

21.5.2 设置 Squid

Squid 很容易安装和配置。因为 Squid 需要空间来保存其高速缓存，所以应该在一台有大量空余内存和硬盘空间的专用机器上运行它。合理的配置是一台带有 2GB 内存和 200GB 磁盘空间的机器。

您可以从您的 Linux 发行版本提供商那里得到 RPM 或者 **apt-get** 格式的 Squid 软件包，或者可以从 www.squid-cache.org 下载 Squid 的新拷贝。如果您选择自己干，那么在解开软件包之后，请在源代码目录树的根目录运行 **configure** 脚本。这个脚本假定您要把包安装在**/usr/local/squid** 中。如果您更喜欢其他的位置，请用**--prefix**=*dir* 选项进行 **configure**。在完成 **configure** 的执行之后，运行 **make all**，然后再运行 **make install**。

一旦您已经安装好了 Squid，必须对配置文件 **squid.conf** 进行本地化设置。您需要对 **squid.conf** 文件的样本做一系列修改，有关情况请参见发布目录中的 **QUICKSTART** 文件。

您还必须手工运行 **squid -z** 来建立并清空保存缓存网页的目录结构。最后，可以用 **RunCache** 脚本手工启动服务程序，您最终将会希望从系统的 **rc** 文件中调用这个脚本，以便在机器引导时启动 Squid 服务程序。

为了测试 Squid，请把桌面 Web 浏览器配置为使用 Squid 服务程序作为代理。这个选项通常可以在浏览器的配置参数中找到。

21.6 匿名 FTP 服务程序的设置

FTP 是 Internet 上最古老和最基本的服务之一，而今天它仍然在广泛使用。虽然 FTP 在一个网点的内部有各种用途，但在 Internet 上最常见的用途就是“匿名 FTP”，它允许在站点上没有账号的用户下载由您提供的文件。

FTP 对发布缺陷修补、软件、文档草稿等类似的东西都是很有用的，但是现在除了 FTP 服务器，没有 HTTP 服务器不能替代的了。钟爱 FTP 服务器的理由相当勉强：FTP 比较可靠一点儿，用户不需要 Web 浏览器就能访问 FTP 站点（尽管他们需要一个 FTP 客户端软件）。

只有要求匿名访问的时候才用普通的 FTP。对于非匿名的应用来说，应该使用更安全的 SFTP。FTP 用明文传输口令，历史上的安全问题一直不断。

ftpd 由 **inetd** 管理，因此在**/etc/inetd/conf** 和**/etc/services** 文件中必须有一个配置项（如果您的系统使用的是 **xinetd** 而不是 **inetd**，那么需要在**/etc/xinetd.d** 目录下为 **ftpd** 创建一个文件，而不是创建一个**/etc/inetd.conf** 配置项）。当一个 FTP 用户匿名登录时，**ftpd** 执行一次 **chroot** 系统调用（是“change root[改变根目录]”的缩写），使**~ftp** 目录之外的文件不可见，也不可访问。因为匿名 FTP 的公开性，所以正确配置 **ftpd**，使得敏感文件不会让外界意外下载就成了一个很重要的问题。

为了在本站点上允许匿名 FTP，可采取下面的步骤。

- 把用户“ftp”添加到普通和掩蔽（shadow）的口令文件中（除了 Debian 之外，其他 Linux 发行版本都应该已经有 ftp 用户）。没有人要以 ftp 账号登录系统，所以用一个“x”作为 ftp 的口令。把**/sbin/nologn** 或者**/bin/false** 作为 ftp 的登录 shell 也是一个好主意。
- 如果没有 ftp 的主目录，就创建它。
- 在**~ftp** 目录下创建子目录 **bin**、**etc**、**lib** 和 **pub**。因为匿名 FTP 会话会运行 **chroot** 把根目录改为**~ftp**，所以子目录 **bin** 和 **etc** 必须提供 **ftpd** 所需命令和配置信息的所有副本。**chroot** 过之后，**~ftp/bin** 和**~ftp/etc** 将伪装成**/bin** 和**/etc**。
- 把 **ls** 程序复制到**~ftp/bin** 目录下。为了增加安全性，将**~ftp/bin/ls** 的模式设置为 111，使它只

www.TopSage.com

能执行。这种调整能防止客户端软件把二进制程序拷走，并研究它的弱点。

- 把 **ls** 所需的共享库复制或者硬链接到**~ftp/lib** 下。检查您的发行版本所带的文档，确定需要哪些文件。注意，硬链接只能在文件处于同一个磁盘分区的时候才能用。
- 把**/etc/passwd** 和**/etc/group** 复制到**~ftp/etc** 下。
- 按照下面的介绍编辑 **passwd** 和 **group** 文件。**ftpd** 只使用 **ls** 命令，还有来自**~ftp/etc** 的 **/etc/passwd** 和**/etc/group** 的文件框架副本。**~ftp** 下的 **passwd** 文件只应包含用户 root、daemon 和 ftp，您必须用“x”作为口令。
- 给**~ftp** 下的文件和目录设置适当的权限。我们推荐按照表 21.4 来设定权限。

表 21.4 ~ftp 下文件和目录推荐的权限

文件/目录	属 主	模 式	文件/目录	属 主	模 式
~ftp	root	555	~ftp/etc/passwd	root	444
~ftp/bin	root	555	~ftp/etc/group	root	444
~ftp/bin/ls	root	111	~ftp/pub	root	755
~ftp/etc	root	555	~ftp/lib	root	555

- 编辑**/etc/ftpusers**，删除“ftp”和“anonymous”两行，让匿名用户能够登录。
- 把您愿意提供的文件放在**~ftp/pub** 中。

匿名 FTP 最大的安全风险之一源于允许用户在 FTP 目录里上载文件。所有人都有写权限的目录，不管它有多隐蔽，都会很快成为交换软件的黑客和孩子们保存文件的“巢穴”，这会耗尽您的带宽，让您置身于一系列您或许并不乐意（如果不完全违法的话）的活动之中。不要出这样的问题，决不要让您系统上的匿名 FTP 目录可以上载东西。

21.7 习题

★E21.1 在您的工作站上配置一个虚拟接口。在配置前后运行 **ifconfig** 命令，了解发生了什么变化。您可以从同一子网里的另一台机器上 ping 到这个虚拟接口吗？从不同的网络上呢？为什么能？或者为什么不能？（要求有 root 权限）

★E21.2 用一个包探测器（**tcpdump**）捕获一个上载信息的双向 HTTP 会话（例如，填写一份表单或者一个搜索字段）。解释这次会话，说明您的浏览器是怎样向 Web 服务器传送信息的。（要求有 root 权限）

★E21.3 用一个包探测器捕获您打开一个繁忙的 Web 网页（比如 amazon.com 或者 cnn.com）时的流量。打开了多少个独立的 TCP 连接？是谁发起它们的？这是使用 TCP 协议最有效的方式吗？（要求有 root 权限）

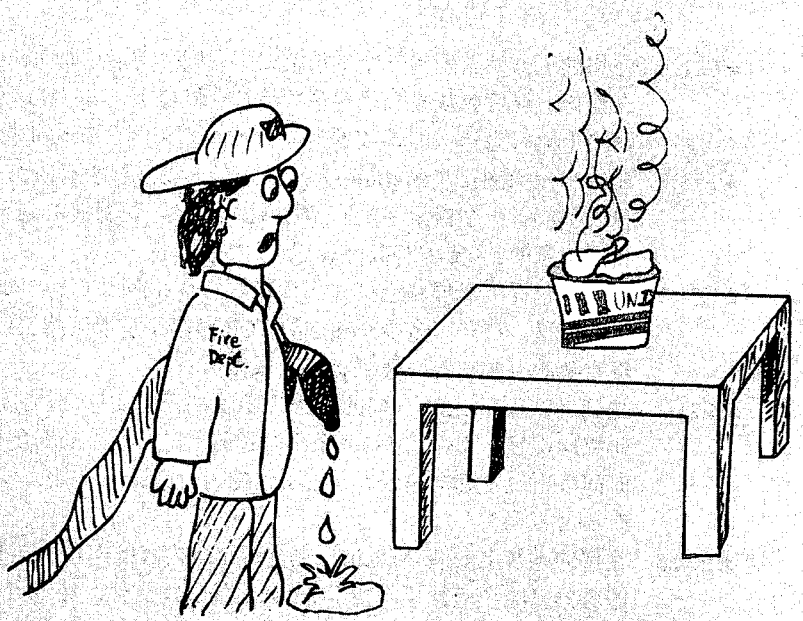
★E21.4 找到可以从 Internet 访问的 Web 服务器的日志文件，或许就是您的站点的主服务器。研究这些日志文件。您对几个小时长的一段时间里的访问模式有什么看法？在那段时间里出现了哪些错误？日志文件的内容暴露出了什么样的隐私问题？（可能要求有 root 权限）

★★E21.5 在您的系统上安装 Apache，创建两个有内容的网页。从其他机器上核实您的 Web 服务器已经在运行了。找出 Apache 的日志文件，让您可以看到哪些浏览器访问了您的服务器。配置 Apache 将它的一些内容网页通过 E21.1 里创建的虚拟接口提供服务。（要求有 root 权限）

www.TopSage.com

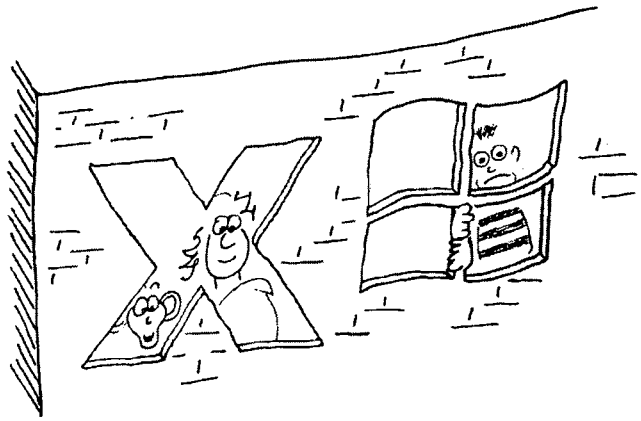
第三部分

其他管理技术



www.TopSage.com

第 22 章　X 窗口系统



X 窗口系统（X Window System）也叫做 X11 或者简称为 X，它是 UNIX 和 Linux 上大多数图形用户环境的基础。X 是一种叫做 W（信不信由你）的窗口系统的自然延续，W 是在 20 世纪 80 年代初作为 MIT 的 Athena 项目的一部分开发的。X 窗口系统的第 10 版于 1985 年发布，这个版本第一次获得了广泛使用，因此后面很快就有了第 11 版（X11）。幸亏这个系统的许可证条款相当慷慨，所以 X 迅速传播到其他平台上，并且出现了多种实现。

1988 年成立了 MIT X Consortium（MIT X 协会），它为 X 协议设定了总体方向。在这之后的 10 年里，这个组织及其后继者持续地发布了一系列对协议的更新。现在的 X11R7.1 是最新和最好的版本，其发展趋势明显是向着这样的方向发展，即让更多的成员加入到软件版本的设计中来，而不是做大现有的成员。

XFree86 后来成为 Linux（以及许多别的平台）事实上的标准 X 服务器实现，到了 2004 年，XFree86 在许可证上的变化促使许多 Linux 发行版本换到了一种 XFree86 的分支上去，它不受新的许可证条款的约束。这个分支由非营利性的 X.Org Foundation（X.org 基金会）维护，是现如今 Linux 上占据主导地位的 X 实现。此外，X.Org 服务器还被移植到 Windows 上，供兼容 Linux 的环境 Cygwin 使用。（Windows 也有几种商业的 X 服务器，参见 26.2.1 节了解更多信息。）

本章介绍 X.Org 版的 X，我们举例的所有 Linux 发行版本都使用它。X.Org 的实现和 XFree86 在体系结构上不一样，但是在系统管理上的大部分细节都相同。经常可以把命令和文件名中的“xorg”换为“xf86”，这样就能猜出 XFree86 下的相应版本。随着时间推移，XFree86 正在逐渐不被使用，将来在这方面就不用再讨论它了。

www.TopSage.com

X 窗口系统可以分成几个关键组件。首先，它提供了一个显示管理器，其主要任务就是验证用户的身份、让用户登录进来，并且从引导脚本启动一个初始的环境。显示管理器也会启动 X 服务器，X 服务器定义了给系统的位图式显示和输入设备（例如，键盘和鼠标）的一个抽象接口。引导脚本还要运行一个窗口管理器，让用户移动、缩放、最大化和最小化窗口，并且管理独立的虚拟桌面。最后，在最低一层，应用软件被链接到一个 widget 库，widget 库实现了上层的用户界面机制，如按钮和菜单。图 22.1 显示了显示管理器、X 服务器和客户端应用之间的关系。

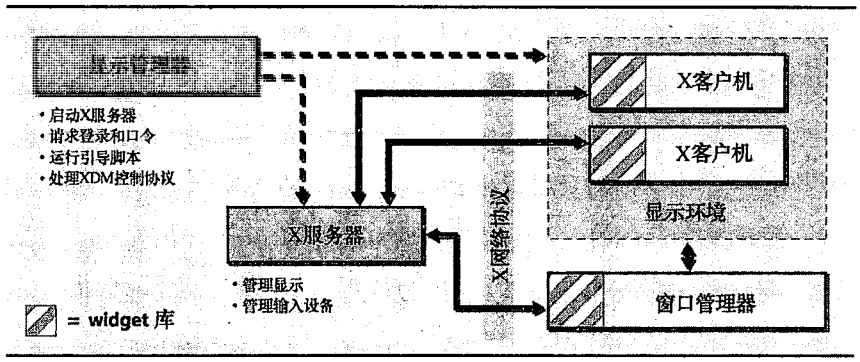


图 22.1 X 的客户机/服务器模型

X 服务器只能理解一套非常基本的绘图原语，这套原语通过一个网络 API 调用，它没有定义高级实体（如按钮、文本框、菜单和滚动条）的编程接口。这样的设计达到了两个重要目标：首先，它能让 X 服务器运行在一个与客户端应用程序完全独立的计算机上；其次，它能让服务器支持各种不同的窗口管理器和 widget 库。

应用程序开发人员可以从几种常用的 widget 库和用户接口标准中自行选择。遗憾的是，这种选择经常要取决于宗教信仰而不是任何切实的设计原因。尽管自由选择的方式挺不错，但是 X 的用户界面不确定性必然导致用户界面在好多年里一直很糟糕。

在这一章里，我们阐述了如何在一个远处的显示器上运行程序，如何启用身份验证功能。接下来，我们讨论如何配置 X.Org 服务器，如何排查配置上的错误。最后，我们简要介绍几种窗口管理器和桌面环境。

22.1 X 显示管理器

当一个用户坐在计算机旁的时候，通常他看到的第一样东西就是显示管理器。显示管理器不是非有不可的，许多用户关掉了显示管理器，从文本的主控台或者从他们的**.login** 脚本直接运行 **startx** 来启动 X（**startx** 自身是一个对 **xinit** 程序的封装程序，由 **xinit** 启动 X 服务器）。

另一方面，显示管理器提供了一个对用户友好的漂亮的登录屏幕，而且增加了一些额外的配置。显示管理器能够通过 XDMCP 协议远程登录到其他 X 服务器上。它还能处理显示验证（参见 22.2.2 节）。最早的显示管理器叫做 **xdm**（代表 X 显示管理器，X display manager），但是现在代替它的显示管理器，如 **gdm**（GNOME 显示管理器）和 **kdm**（KDE 显示管理器）既提供了或多或少一样的功能，又有更漂亮的外观。

在典型的场合下，显示管理器启动 X 服务器，验证用户的身份，让用户登录到系统中，并且执行用户的引导脚本。有一组配置文件，其中大部分经常位于**/etc/X11/xdm** 目录下，规定了 **xdm** 将如何运行。例如，如果要在别的虚拟终端上运行多个服务器，您可能想要编辑 **Xservers** 这个文件，改变用于该服务器的显示的数量。或者，如果您已经规定了适用于多个系统的显示布局，那么您可能要用**-layout**

www.TopSage.com

选项改变服务器的布局。

xdm 在启动了 X 服务器之后，就会提示输入用户名和口令。根据**/etc/pam.d/xdm**（如果您使用 KDE 或者 GNOME 显示管理器的话分别是 **kdm/gdm**）中规定的 PAM（Pluggable Authentication Modules，可插入式身份验证模块）来验证用户的口令。登录屏幕也能够显示登录到其他几种桌面环境的选项，其中包括下面会讨论到的重要的 failsafe（故障安全）选项。参见 20.5.6 节了解有关 PAM 的更多知识。

显示管理器最后的任务是执行 shell 脚本 **Xsession**，这个脚本建立起用户的桌面环境。**Xsession** 脚本也往往是在**/etc/X11/xdm** 里，它是一个系统级的引导脚本。它设定应用的默认值、安装标准的键绑定，并且启动窗口管理器、任务栏、帮助程序、可能还有其他一些程序。GNOME 和 KDE 都有它们自己的引导脚本，按照与 GNOME 和 KDE 配置工具一致的方式配置好用户的桌面。比起用户编辑自己的引导脚本来，这种方案更不容易出错。

当执行完**~/.xsession** 之后，用户就退出系统，显示管理器又重新提示输入用户名和口令。

因此，**~/.xsession** 必须在后台启动所有的程序（在每条命令之后加上一个&），对此只有一个例外，一般是窗口管理器（如果**~/.xsession** 里所有的命令都在后台运行，这个脚本也就立即结束了，用户在刚登录之后立即又退出了系统）。有了窗口管理器这个最后的前台进程，用户只有在窗口管理器退出之后，才会从系统退出。

failsafe（故障安全）这个登录选项能让用户登录进来修正其受损的引导脚本。这个选项通常可以从显示管理器的登录屏幕上选择。它只打开一个简单的终端窗口，该窗口一关闭，系统就让用户退出。每个系统都应该有 failsafe（故障安全）选项，它能帮助用户解决困难，于是用户就不会半夜把系统管理员叫来解决问题。

忘记留一个前台运行的进程是常见的启动问题，但是它几乎不是问题的唯一可能。如果问题的原因不明显，那么您可能不得不要参考一下**~/.xession-errors** 这个文件，它里面包含了从**~/.xession** 执行的命令的输出。找找看有没有错误或者其他意外的动作。在紧急情况下，可以把**~/.xession** 整个放到一边，做到不要它您也能登录到系统里。然后再每次恢复一两行配置，直到您找到引发问题的配置行为止。

22.2 运行一个X应用程序

运行一个 X 应用所需要的步骤乍看起来似乎过于复杂了。不过，您很快就会发现客户机/服务器的显示模型所带来的灵活性。因为显示更新是通过网络传输的，所以一个应用（客户机）运行的主机可以和显示其图形用户界面的主机（服务器）完全分开。一个 X 服务器可以从许多不同的应用程序连接上来，所有应用程序都运行在不同的计算机上。

为了让这个模型起作用，必须告诉客户机连接到什么显示，什么屏幕要放到那个显示上。一旦客户机连接上了服务器，它必须让自己的身份在 X 服务器上验证通过，确保坐在显示前面的人有权建立连接。

即使有了身份验证，X 内在的安全性还是相当薄弱。您可以通过 SSH 建立连接，以更以安全的方式管理它们（参见 22.2.3 节）。我们强烈建议在 Internet 上使用 SSH 建立 X 连接。即使用于本地通信也不是不合理。参见 20.11.3 节了解有关 SSH 的更多知识。

22.2.1 环境变量 DISPLAY

X 应用程序查询环境变量 DISPLAY 知道到哪里显示自己。这个变量包括服务器的主机名或者 IP 地址、显示号（标识要连接上的一个具体的 X 服务器实例）、以及一个可选的屏幕号（用于有多个显

www.TopSage.com

示器的情况下的显示)。当应用程序运行的计算机和显示其界面的计算机是同一台机器的时候，为简单起见，您可以省略其中的大部分参数。

下面的例子给出了显示信息的格式和用于设置环境变量的 bash 语法：

```
client$ DISPLAY=servername.domain.com:10.2; export DISPLAY
```

这一设定把 X 应用程序指向了主机 servername.domain.com、显示 10 和屏幕 2。应用程序建立一条连到服务器的 TCP 连接，去连接服务器上端口号为 6000 加上显示号的端口(在本例中是 6010 端口)，处理该显示的 X 服务器应该监听这个端口。

要记住一点，每个进程都有自己的环境变量。当您在一个 shell 里设置了 DISPLAY 之后，只有从那个 shell 里执行的程序才会继承它的值。如果您从一个 **xterm** 里执行上述命令，而从另一个 **xterm** 里运行您最喜欢的 X 应用程序，那么该应用程序不会访问到您精心编排的 DISPLAY 变量。

值得一提的另一点是，尽管 X 应用程序把它们的图形输出送到了专门的 X 服务器上，它们仍然把有本地的 stdout（标准输出）和 stderr（标准出错）通道。有些出错输出仍将出现在 X 应用程序运行的那个终端串口里。

如果客户机和服务器都是本地机构的一部分，那么通常可以在环境变量 DISPLAY 里省略服务器的完整域名，具体取决于名字服务器的解析器是如何配置的。因为大多数系统都只运行一个 X 服务器，所以显示号通常为 0。屏幕号可以省略，省略的时候认定屏幕号为 0。因此大多数时候都把 DISPLAY 的值设为 servername:0。

如果客户端应用程序碰巧和 X 服务器运行在同一台机器上，那么您甚至可以省略主机名，进一步简化 DISPLAY 变量。这个特性可不只是摆摆样子：主机名为空的时候，客户机的库要使用一个 UNIX 域套接口而不是一个网络套接口来联络 X 服务器。除了速度更快、效率更高之外，这种连接方法还绕过了本地系统上任何阻止对外 X 连接的防火墙。于是，环境变量 DISPLAY 可能有的最简单的值就只是一个“:0”。

客户机上读取环境变量 DISPLAY 的那些库通常还可以接受以命令行参数形式提供的这一信息。例如，命令：

```
client$ xprogram -display servername:0
```

等价于把 DISPLAY 设为“servername:0”再运行这个程序。命令行选项会覆盖环境变量的设定。如果您在同一台机器上运行有不同显示的多个程序，那么这一功能就特别方便。

22.2.2 客户机身份验证

虽然一般认为 X 环境相当不安全，但是每一种预防措施都有助于防止未经授权的访问。在安全还未成为如此受关注的问题之前，一台主机只要已经被 **xhost** 命令标为安全，那么 X 服务器就会接受从它来的所有客户机。但是由于那台主机上的任何用户都能连接到您的显示上，干什么都行（要么是有意的，要么是出于糊涂），用 **xhost** 把访问授权给客户机的做法最终被废弃了。我们不再深入讨论它。

提供基于主机安全性最流行的方法称为 magic cookie 验证。虽然 magic cookie 的概念可能会引起一些读者似曾相识的感觉，在这里它们用作验证 X 连接的身份。其基本思想是，X 显示管理器在登录过程开始的时候生成一个随机数，叫做一个 cookie。针对服务器的 cookie 被写入用户主目录下的 **~/.Xauthority** 文件里。知道这个 cookie 的任何客户端程序都被允许连接服务器。

用户可以运行 **xauth** 命令查看已有的 cookie，还可以向这个文件添加新的 cookie。

www.TopSage.com

要展示其如何工作，最简单的方法就是举个例子。假定您已经把客户系统上自己的环境变量DISPLAY设为把X应用显示在您的机器上。不过，当您运行一个程序的时候，会得到下面的这样的报错：

```
client$ xprogram -display server:0
Xlib: connection to "server:0.0" refused by server
xprogram: unable to open display 'server:0'
```

这则出错消息告诉您，客户机没有正确的cookie，所以远程服务器拒绝连接。为了得到正确的cookie，可以登录到服务器上（如果您正试着在它上面显示应用程序，或许您已经登录进去的那台服务器），执行**xauth list**命令列出服务器的cookie：

```
server$ xauth list
server:0  MIT-MAGIC-COOKIE-1  f9d888df6077819ef4d788fab778dc9f
server/unix:0  MIT-MAGIC-COOKIE-1  f9d888df6077819ef4d788fab778dc9f
localhost:0  MIT-MAGIC-COOKIE-1  cb6cbf9e5c24128749feddd47f0e0779
```

服务器上的每个网络接口都有一行。在本例中，我们有一个给以太网的cookie，一个给UNIX域套接口（用于本地连接）的cookie，以及一个给localhost环回（loopback）网络接口的cookie。

把cookie弄到客户机上最简单的办法就是“拷贝-粘贴”（在不使用SSH的情况下，因为SSH会替你协商cookie）。大多数终端模拟程序（例如，**xterm**）都能让您用鼠标选择文本，并粘贴到另一个窗口里去，通常要按鼠标中键。作为**xauth add**命令的输入，也能接受和**xauth list**的显示一样的格式，这很方便。您可以把下面这样的cookie加到客户机上：

```
client$ xauth add server:0 MIT-MAGIC-COOKIE-1
    9d888df6077819ef4d788fab778dc9f
```

您应该在客户机上执行**xauth list**命令核实cookie是否正确地加到了客户机上。在设定了环境变量DISPLAY，并把cookie加到客户机上之后，现在应用程序就能正确地在服务器上显示了。

如果您在让cookie起作用的过程中遇到了问题，可以临时退回到**xhost**验证方式，核实一下没有别的问题出现（例如，防火墙或者本地网络限制不让客户机访问那台服务器）。在测试完成后，一定要运行**xhost -**（也就是说，**xhost**只带一短线作为参数）禁用**xhost**验证方式。

22.2.3 用SSH转发X连接

Magic cookie提高了安全性，但它们几乎做不到万无一失。任何人，如果能获得您的系统显示的cookie，都能连上系统显示，运行程序监视您的活动。即使没有您的cookie，X协议通过网络传输的数据不加密，实际上也能让任何人监听到。

您可以用SSH（secure shell protocol）提高安全性。SSH提供了一种身份验证和加密终端服务。不过，SSH也能通过一条安全隧道转发包括X协议数据在内的任何网络数据。X转发和普通的SSH端口转发很相似，但是因为SSH知道X，所以您可以获得一些附加功能，包括在远程机器上的伪显示和magic cookies的协商传输。参见20.11.3节了解有关SSH的更多知识。

您一般要从运行X服务器的机器通过**ssh**登录到想要运行X应用程序的机器上。这种安排可能读起来有点儿让人糊涂，因为SSH客户机和X服务器运行在同一台机器上，它连接到的SSH服务器和X客户应用程序在同一台机器上。让情况更乱的是，SSH为您的X服务器建立的虚拟显示对于远程系统来说是在本地。

图22.2画出了X流量如何通过SSH连接。

www.TopSage.com

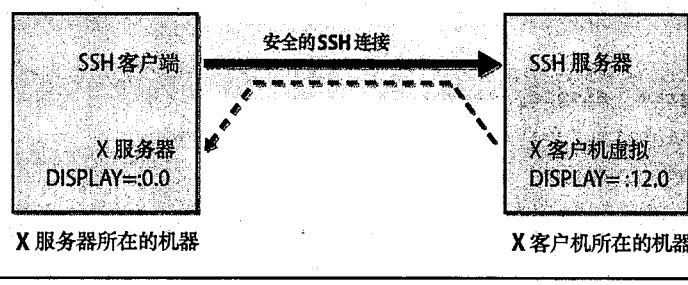


图 22.2 使用 SSH 的 X

DISPLAY 变量和验证信息由 **ssh** 自动设置。显示号从:10.0 开始，每条转发 X 流量的 SSH 连接加 1。用一个例子有助于说明这个过程：

```
x-server$ ssh -v -X x-client.mydomain.com
OpenSSH_3.9p1, OpenSSL 0.9.7a Feb 19 2003
debug1: Reading configuration data /home/boggs/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to x-client.mydomain.com [192.168.15.9] port 22.
debug1: Connection established.
Enter passphrase for key '/home/boggs/.ssh/id_rsa':
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
debug1: Entering interactive session.
debug1: Requesting X11 forwarding with authentication spoofing.
debug1: Requesting authentication agent forwarding.
x-client$
```

您可以从最后两行看到，客户机请求转发 X11 的应用程序流量。必须在 SSH 服务器和 SSH 客户端都启用 X 转发功能，客户机依然必须要有服务器的正确 cookie。如果工作似乎显得不正常，那么像上面那样试试**-X** 和**-v** 标志（适用于 OpenSSH）来显式地启动 X 转发功能，并要求详细的输出。还要检查**/etc/ssh** 下的 SSH 全局配置文件，确保在系统管理上没有禁止 X11 转发功能。一旦能登录进入系统，您就可以检查您的显示和 magic cookie：

```
x-client$ echo $DISPLAY
localhost:12.0
x-client$ xauth list
x-client/unix:12  MIT-MAGIC-COOKIE-1  a54b67121eb94c8a807f3ab0a67a51f2
```

注意，DISPLAY 指向了 SSH 服务器上的一个虚拟显示。其他 SSH 连接（从您和从其他用户来的连接）都分配了不同的虚拟显示号。有了设好的 DISPLAY 和 cookie，现在您就可以运行客户端的应用程序了。

```
x-client$ xeyes
debug1: client_input_channel_open: ctype x11 rchan 4 win 65536 max 16384
debug1: client_request_x11: request from 127.0.0.1 35411
debug1: channel 1: new [x11]
debug1: confirm x11
debug1: channel 1: FORCE input drain
```

有了 **ssh -v** 提供的调试信息，您就能看到 ssh 已经接收到了 X 连接请求，并且尽责地把它转发给 X 服务器。在长距离链路上，转发会有点儿慢，但是应用程序应该能最终出现在您的屏幕上。

www.TopSage.com

22.3 配置 X 服务器

X.Org 的服务器叫做 **Xorg**，对于一定的硬件环境，它是出了名的难配置。但它落得这么个名声不是没有道理的。一部分原因在于，**Xorg** 要支持的大量图形硬件、输入设备、视频模式、分辨率和颜色深度导致了它的配置复杂。在早期的 XFree86 上，一个新用户经常会受不了晦涩难懂的配置文件，这个文件里的内容似乎都是随机的数字，它们源自于显示器并不存在的手册背后的隐晦信息。**Xorg** 的配置文件为这些看似随机的数字提供了更好的结构。

Xorg 的配置文件一般可以在**/etc/X11/xorg.conf** 下找到，但是 X 服务器会搜索整个目录树来找它。它的手册页提供了一份完整的清单，但是有一点要指出，有些搜索 **Xorg** 的路径包括主机名和一个全局变量，能让您更容易在一个中央位置为多个系统保存配置文件。

有几个程序能够帮助您配置 X（例如，**xorgconfig**），但是理解配置文件的结构，以防万一您需要直接查看或者编辑配置，始终是个不错的想法。您可以运行 **Xorg -probeonly** 命令，查看输出的显卡芯片组和其他探测到的值，直接收集到一些有用处的初始信息。您可以运行 **Xorg -configure** 命令让服务器根据探测到的值创建一个初始的配置文件。如果您其他什么都没有的话，这个文件是个不错的开端。

xorg.conf 文件的内容被组织成几段，每一段配置以关键字 `Section` 开头，以 `EndSection` 结尾。表 22.1 列出了最常用的配置段。

表 22.1 xorg.conf 文件的配置段

配置段	说 明
`ServerFlags`	列出一般性的 X 服务器配置参数
`Module`	指定为加速图形显示、平滑字体等功能而动态加载的扩展模块
`Device`	配置显卡、驱动程序和硬件信息
`Monitor`	描述显示器的物理参数，包括刷新率和显示分辨率
`Screen`	把一台显示器和一个显卡（Device）联系起来，规定在该配置中能采用的分辨率和颜色深度
`InputDevice` `ServerLayout`	把输入设备绑定到一组屏幕上，并且定出屏幕彼此之间的相对位置

先规定输入和输出设备的配置段，然后再把它们组合为多种不同的布局，从后往前构造一个配置文件是最简单的方法。以这种分层次的方法，一个简单的配置文件可以用于许多个 X 服务器，每一个只是在硬件上有所不同。对于一个有多块显卡和多个显示器的系统来说，这也是一种合理的方法。

图 22.3 给出了其中一些配置段如何一起构成了 X.org 的配置层次。一台物理上的显示器（`Monitor`）加一块显卡（`Device`）就构成一个屏幕（`Screen`）。一组屏幕（`Screen`）再加上若干输入设备（`InputDevice`）就构成了一个服务器布局（`ServerLayout`）。尽管对于一个 **Xorg** 进程的实例只能有活动的服务器布局，但在一个配置文件里可以定义多个服务器布局。

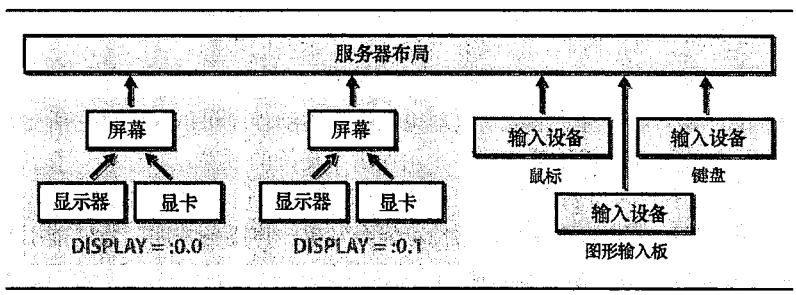


图 22.3 xorg.conf 配置文件中各部间的关系

www.TopSage.com

构成 **xorg.conf** 文件的一些配置段相当固定。默认的设置往往就可以直接供一个现有或者举例的配置文件使用。其他配置段，如 Device、Monitor、Screen、InputDevice 和 ServerLayout 段则取决于主机的硬件设置。我们在接下来的几小节里详细讨论这些配置段最有意思的地方。

22.3.1 Device 段

一个 Device 段描述一块特定的显卡。您必须提供一个字符串来标识这块显卡，还要给它提供了一个正确的驱动程序。只有当相应的 Section 段用到这个设备的时候，才加载这个驱动程序。典型的 Device 段可能像下面这个样子：

```
Section "Device"
    Identifier      "STBVirge3D"
    Driver          "s3virge"
    option          value
    ...
EndSection
```

在本例中，驱动程序的说明书说明了它能驱动的硬件以及该驱动程序支持的选项。如果您遇到了奇怪的显卡，您可能要试着通过设置选项，关闭硬件加速功能（如果有的话）、降低访问显存的速度或者修改 PCI 参数。总的来说，在您开始随便乱改参数值之前，上网查查别人是否已经遇到过类似问题，一般是个不错的做法。

22.3.2 Monitor 段

Monitor 段描述连接到计算机的显示器。它能够指定详细的刷新率。对于比较老的硬件来说，需要有刷新率的信息，但是大多数现在的显示器都可以通过探测得到这个值。显示器的规格通常都能从制造商的 Web 网站上获得，但是什么都比不上显示器原配的说明书。无论哪一种方法，您都要至少知道该显示器型号的水平同步频率和垂直刷新频率。

典型的 Monitor 段可能像下面这个样子：

```
Section "Monitor"
    Identifier    "ViewSonic"
    Option        "DPMS"
    HorizSync     30-65
    VertRefresh   50-120
EndSection
```

和所有配置段一样，Identifier 行给显示器分配一个名字，以后您可以用这个名字来代表这台显示器。我们在这里打开了 DPMS（Display Power Management Signaling，显示器电源管理），所以当我们偷偷离开去吃点心喝咖啡的时候，X 服务器会关闭显示器的电源。

HorizSync 和 VerRefresh 两行应该填上您显示器的正确值。这两个值可以用频率范围（如上所示）或者逗号隔开的多个频率值来指定。从理论上说，驱动程序可以探测到它所支持的显示器型号的值，但是指定这两个参数可以避免驱动程序企图使用不支持的频率。

早期 CRT 显示器有不少因为调为不正确的频率去运行而被损坏的吓人故事，但是现如今的 CRT 适应能力似乎强了一点儿。在最坏情况下，它们可能会发出一声高频啸叫，肯定能引起小狗的注意。现代的 LCD 显示器对信号波动的忍耐力更强，但是在试验显示器的频率时，做到投鼠忌器，可能仍然不失为明智之举。要准备好，如果显示器接受不了它收到的信号，就关掉它。

www.TopSage.com

22.3.3 Screen段

Screen段把一个设备（显卡）联系到一台显示器，这台显示器有特定的颜色深度和特定的一组显示分辨率。下面的例子使用了上面提到的显卡和显示器。

```
Section "Screen"
    Identifier     "Screen 2"
    Device         "STBVirge3D"
    Monitor        "ViewSonic"
    DefaultDepth 24
    Subsection "Display"
        Depth    8
        Modes    "640x400"
    EndSubsection
    Subsection "Display"
        Depth    16
        Modes    "640x400" "640x480" "800x600" "1024x768"
    EndSubsection
    Subsection "Display"
        Depth    24
        Modes    "1280x1024" "1024x768" "800x600" "640x400" "640x480"
    EndSubsection
EndSection
```

不出您所料，有一行 Identifier 配置了屏幕的名字，并且提到了前面定义的显卡和显示器的标识符。在我们介绍的配置段里，第一次见到了配置子段。为每一种颜色深度定义了一个子段，默认子段用 DefaultDepth 指定。

一个X服务器的实例只能运行在颜色深度上。X服务器在引导的时候，要判断该颜色深度所支持的分辨率。分辨率可能有哪些取决于显卡上显存的数量。比较老的显卡显存也少，一般来说，在颜色深度大的时候，分辨率会受到限制。

X的特殊键盘组合（在22.4.1节介绍）描述了如何在这里定义的分辨率之间循环切换。

任何一款好的现代显卡都应该能让您的显示器在24位或者32位色的情况下达到其最大分辨率。如果您想要运行的老程序要求服务器运行在8位色，那么可以在另外的虚拟控制台上运行第二个X服务器。在**Xorg**的命令行上使用**-depth 8**标志就能覆盖 DefaultDepth 选项。

22.3.4 InputDevice段

InputDeivce 段描述了诸如键盘或者鼠标这样的输入事件来源。每个设备都有自己的 InputDevice 段，而且和其他配置段一样，每个 InputDeivce 段也要用 Identifier 配置名字。如果您让多台有着不同硬件的机器共享一个配置文件，那么可以定义所有的输入设备，但是只使用在 ServerLayout 段引用的那些设备。下面是一个典型的键盘定义：

```
Section "InputDevice"
    Identifier     "Generic Keyboard"
    Driver         "Keyboard"
    Option         "AutoRepeat" "500 30"
    Option         "XkbModel" "pc104"
    Option         "XkbLayout" "us"
EndSection
```

www.TopSage.com

您可以在键盘的定义中设定选项，说明您对 Control 和 Cap Lock 键正确位置的特殊要求，此外还可以配置别的。在本例中，AutoRepeat 选项指出一个键需要按下多长时间才开始重复，以及重复的速度有多快。

鼠标在另一个 InputDevice 段里配置：

```
Section "InputDevice"
    Identifier    "Generic Mouse"
    Driver        "mouse"
    Option        "CorePointer"
    Option        "Device" "/dev/input/mice"
    Option        "Protocol" "IMPS/2"
    Option        "Emulate3Buttons" "off"
    Option        "ZAxisMapping" "4 5"
EndSection
```

CorePointer 选项把这个鼠标指定为系统的首要指示设备。和这个鼠标关联的设备文件作为一个 Option 来指定，它一般设为**/dev/input/mice**，它是鼠标设备的多路复用文件。protocol 取决于使用的鼠标品牌，您可以把它设为 auto，这样服务器会尝试替你检测到。如果您的鼠标滑轮不工作，那么试试把这个 protocol 改为 IMPS/2。如果有好几个按钮，那么可能需要使用 ExplorerPS/2 这个 protocol。

如果**/dev/input/mice** 对您的鼠标不起作用，那么配置会稍微复杂一些。**gpm** 程序在文本模式的虚拟终端里，实现了类似 X 的、能用鼠标控制剪切和粘贴的功能。不过，一次只能有一个程序可以打开一个传统的鼠标设备。为了解决这个问题，**gpm** 把鼠标数据复制到一个 FIFO 文件里，于是像 X 服务器这样的应用也能看到它了（通过把这个 FIFO 当作鼠标设备）[1]。这种方案在把 **gpm** 功能加入到系统的同时又能让它保持透明。

例如，下面的命令让 gpm 从**/dev/mouse** 以 IMPS/2 协议获得其输入，并不必做协议转换就把它转发给**/dev/gpmdata**（名字不能配置）这个 FIFO 文件。

```
$ gpm -m /dev/mouse -t imps2 -Rraw
```

接下来，您要把 **xorg.conf** 文件里的鼠标设备文件改为**/dev/gpmdata**。因为 **gpm** 必须在启动 X 服务器之前运行起来，所以必须把这条加进诸如**/etc/init.d/gpm** 这样的系统引导脚本中。参见 2.5 节了解有关引导脚本的更多知识。

Emulate3Buttons 这个选项通过定义同时点鼠标双键等于点鼠标中建的方式，让一个双键鼠标去模拟一个三键鼠标。有时候需要 ZAxisMapping 选项，通过适当映射鼠标键的方法，支持滚动轮或者游戏杆设备。现如今的大多数鼠标都至少有两个键、一个滚轮[2]。

22.3.5 ServerLayout 段

ServerLayout 段是层次型配置的最高层节点。服务器将要在上面运行的每种硬件配置都应该有自己的 ServerLayout 配置段。这个由某个特定 X 服务器使用的布局通常要在服务器的命令行上指定。

下面是一个完整的 ServerLayout 段的例子：

[1] 用 **mknod** 命令可以创建 FIFO 文件。例如，**mknod p /dev/gpmdata**。

[2] 不是所有可能的配件 **Xorg** 都支持。有些配件是单独销售的。

www.TopSage.com

```
Section "ServerLayout"
    Identifier     "Simple Layout"
    Screen         "Screen 1" LeftOf "Screen 2"
    Screen         "Screen 2" RightOf "Screen 1"
    InputDevice    "Generic Mouse" "CorePointer"
    InputDevice    "Generic Keyboard" "CoreKeyboard"
    Option         "BlankTime"    "10"  # Blank the screen in 10 minutes
    Option         "StandbyTime"  "20"  # Turn off screen in 20 minutes (DPMS)
    Option         "SuspendTime"  "60"  # Full hibernation in 60 minutes (DPMS)
    Option         "OffTime"      "120"# Turn off DPMS monitor in 2 hours
EndSection
```

这段配置把其他所有配置段都结合到一起，表示一个 X 显示。它首先要求有 Identifier，给这个特定的布局起个名字。然后把一组屏幕和这个布局关联起来[3]。如果多个显示器分别连接到不同的显卡上，那么由可选的指令指定的每个屏幕就指出了它们在物理上是怎样安排的。在本例中，屏幕 1（Screen 1）在左边，而屏幕 2（Screen 2）在右边。

有些显卡能够一次驱动多台显示器。在这种情况下，在 ServerLayout 段只能指定一个 Screen。对于 NVIDIA 显卡来说，目前这种应用在 Linux 上最常见的做法是，您在 Driver 段设定了一个选项，标明支持 TwinView。这个配置的细节超出了本书的范围，但在各种 Web 论坛里很容易就能找到说明。

屏幕列表的后面是与这个布局关联的一组输入设备。CorePointer 和 CoreKeyboard 选项会传给 InputDevice 段，指出要为这个配置激活的设备。那些选项还可以直接在其相应的 InputDevice 段里设定，但是在 ServerLayout 段里设定它们会更清楚。

最后几行配置了几个专门针对布局的选项。在上例中，这些选项都和 DPMS 有关系。DPMS 是一个接口，它告诉符合能源之星的显示器何时自行关闭电源。显示器还必须在其相应的 Monitor 段里启用 DPMS 选项。

22.4 故障排查和调试

在过去的 10 年里，X 服务器的配置走过了一条漫长的道路，但要一切能按照您所喜欢的方式工作，仍然比较困难。您可能需要做些实验，才能确定显示器的频率、驱动程序的选项、专有的驱动程序或者是否有三维渲染扩展功能。颇具讽刺意味的是，当您对看到屏幕上的调试结果最感兴趣的时候，正好也是显示器不能正确工作的时候。还好，X.Org 服务器提供了跟踪问题所需要的全部信息。

22.4.1 X 的特殊键盘组合

因为 X 服务器接管了您的键盘、显示器、鼠标和社会生活，所以不难想象，如果出了问题不能干活了，您除了关掉系统电源别无他法时该多么无奈。不过，在达到这样的程度之前，还是可以尝试一番的。如果您按下 Control 和 Alt 键，再同时按一个功能键（F1 到 F6），X 服务器会把您带回到一个基于文本的虚拟终端。您可以从那里登录到系统中，调试问题。要返回运行在虚拟终端 7 上的 X 服务器，可以按<Alt-F7>[4]。如果您连着网络，在按机器上的 reset 按钮之前，也可以试着从另一台计算机登录进来，杀死 X 服务器程序。

[3] 回忆一下，屏幕表示一个显示器/显卡在一个特殊颜色深度上的组合。

[4] X 服务器要求一起按<Control>键和<Alt-Fn>组合键来切换虚拟终端，但是文本的主控台不是这样。

www.TopSage.com

如果监视器和显卡的视频信号不同步，那么就要试着修改屏幕的分辨率。配置文件里 Screen 段的 Mode 行规定了可以使用的分辨率。要激活哪个 Mode 行算正确则取决于颜色深度，参考 22.3.3 节了解详情。X 服务器默认使用激活的 Mode 行里出现的第一个显示分辨率，但是您可以通过按下 Control 和 Alt 键，再按数字小键盘上的加号（+）或者减号（-）键，在不同的分辨率之间循环设定。

按下<Contrl-Alt-Backspace>会立即杀死 X 服务器。如果您从一个主控台运行了 X 服务器，当服务器退出的时候，您就发现自己又返回到了主控台。如果是一个显示管理器启动了这个 X 服务器，那么它通常会派生出一个新的服务器，再次提示输入登录名和口令。您必须从一个文本的主控台杀死这个显示管理器（**xdm**、**gdm** 等），才能让它停止派生新的 X 服务器。

22.4.2 正常 X 服务器出问题

一旦您重新获得了对机器的控制权，您可以开始追踪问题的原因。最简单的入手处就是 X 服务器的输出。这些输出偶尔能在虚拟终端 1 上看到（<Control-Alt-F1>），引导程序的输出也都送到这个终端。最常见的情况是，X 服务器的输出被记录在了**/var/log/Xorg.0.log** 这样的日志文件里。

从下面可以看到，每一行开头都有一个特征分类符号。您可以使用这些特性符号来找到出错（**EE**）和警告（**WW**），还能判断出服务器是怎样找到每一种信息的：通过默认配置（==）、在一个配置文件里（**）、自动检测到的（--）或者由 X 服务器的命令行指定的（++）。

让我们研究一下下面的片段：

```
X Window System Version 6.8.2
Release Date: 9 February 2005
X Protocol Version 11, Revision 0, Release 6.8.2
Build Operating System: Linux 2.4.21-23.ELsmp i686 [ELF]
Current Operating System: Linux chinook 2.6.12-1.1372_FC3 #1 Fri Jul 15 00:59:
    10 EDT 2005 i686
Markers: (--) probed, (**) from config file, (==) default setting,
      (++) from command line, (!!) notice, (II) informational,
      (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Mon May  1 08:41:02 2006
(==) Using config file: "/etc/X11/xorg.conf"
(==) ServerLayout "Default Layout"
(**) |-->Screen "Screen0" (0)
(**) |   |-->Monitor "Monitor0"
(**) |   |-->Device "Videocard1"
(**) |-->Input Device "Mouse0"
(**) |-->Input Device "Keyboard0"
```

最前面的几行告诉您 X 服务器的版本号以及它所实现的 X11 协议的版本号。后面的几行告诉您这个服务器在日志文件位置、配置文件位置和获得服务器布局上，都使用了默认值。从配置文件获得的显示器和输入设备都按结构回显出来。在日志中常见的一个问题就是难以使用某些屏幕分辨率，这类问题通常的症状为，那些显示分辨率不能工作，或者 X 服务器输出一些出错消息，如“Unable to validate any modes; falling back to the default mode（任何模式都无效；返回默认模式）”如果您没有给显示器指定一系列频率，那么 X 服务器会使用 EDID（Extened Display Identification Data，扩展的显示器识别数据）探测它们。如果您的显示器不支持 EDID，或者如果您的显示器在 X 启动的时候被关闭了，那么你就需要在配置文件里的 Monitor 段设定 X 要用的频率范围。

从 EDID 探测获得结果有舍入误差，这种误差可以造成某些分辨率没有出现，尽管显卡和显示器都应该支持这些分辨率。诸如“No valid modes for 1280x1024; removing（没有用于 1280×1024 的有效模式；删除）”这样的日志信息就预示着出现了这样的问题。解决办法是告诉 X 服务器，忽略 EDID

www.TopSage.com

大学网 TopSage.com

的信息，使用通过下面的 Device 段配置行所指定的频率：

```
Option   "IgnoreEDID" "true"
Option   "UseEdidFreqs" "false"
```

再举个例子，假定您忘记正确地定义鼠标的配置段。那么在输出里会出现类似下面的错误信息：

```
(==) Using config file: "/etc/X11/xorg.conf"
Data incomplete in file /etc/X11/xorg.conf
        Undefined InputDevice "Mouse0" referenced by ServerLayout "Default
    Layout".
(EE) Problem parsing the config file
(EE) Error parsing the config file
Fatal server error:
no screens found
```

X 启动并运行，您已经登录到系统里之后，您就可以运行 **xdpyinfo** 命令获得更多有关 X 服务器配置的信息[5]。**xdpyinfo** 命令的输出再次告诉您显示的名字以及 X 服务器的版本信息。它还告诉您可以有的颜色深度、已经加载的扩展，以及已经定义的屏幕及其分辨率和颜色配置。**xdpyinfo** 的输出可以用一个脚本（诸如您的**~/.xsession** 文件）进行分析，判断出当前活动屏幕的尺寸，并设置正确的桌面参数。

对于调试工作来说，**xdpyinfo** 最有用的地方是能判断 X 服务器是否在运行并且监听网络查询，它是否配置了正确的屏幕和分辨率，以及是否正运行在所期望的颜色位深度下。如果这一步能奏效，那么您就准备好可以启动运行 X 应用程序了。

22.5 桌面环境简述

多年以来，X 的客户机/服务器模型所具有的灵活性和简单性让 widget 库、窗口管理器、文件浏览器、工具栏程序以及工具程序有了蓬勃的发展。在 MIT 那个产生了 X 的 Athena Project 上，还产生了 Athena widget 和 twm（Tom's Window Manager，得名于它的编写者 Tom LaStrange，它也叫做 Tab Window Manager）。这些基础性的工具构成了早期 X 应用事实上的标准。

Sun 公司和 AT&T 联合开发的 OpenLook 是另一种工具集，它引入了椭圆按钮和图钉，让人们能够看到菜单和对话框。大约与此同时，Open Software Foundation（开放软件基金会）带来了颇有竞争力的 Motif 平台（后来叫做 CDE，即 Common Desktop Environment，公共桌面环境），Sun 公司最后也采用了 Motif。在当时，这些工具集的三维显示效果很漂亮，而高级 UI 元件（如滚动条、菜单）的流行则使之成为新软件的理想选择。不过，这两种工具集都完全是专有的技术，开发库和窗口管理器的许可证费也让公众难以接触到它们。

随着开放源代码的操作系统（如 Linux）的出现，也出现开放源代码的桌面环境。FVWM（"F" Virtual Window Manager）因其高度可配和支持"虚拟桌面"的特点而在 Linux 上流行起来，虚拟桌面扩大了用户的有效工作区域，超出了当时大多数 PC 上的低分辨率显示器的显示范围[6]。但是没有与之关联的 widget 库，所以用户仍然要面对大量的程序，每个程序都有其不同的显示

[5] 我们不推荐以 root 身份登录到 X 里，因为这样的做法会在 root 的主目录下（通常是/）创建一堆默认的引导文件。这样非常不安全。相反，以一个普通用户身份登录，然后使用 **sudo**。Debian 和 Ubuntu 默认就是这么要求的。

[6] 实际上，FVWM 非常灵活，灵活到能把它的外观配置成像 **twm** 或者 **mwm**（Motif Window Manager）一样。

www.TopSage.com

风格。

随着应用程序越来越高级，对更高级的用户界面功能要求也越发强烈，显然必须采用一种更通用的方法把用户的使用经验统一起来，并且给应用软件开发人员提供更好的支持。从这样的需求里诞生了两种重要的现代 Linux 桌面环境：GNOME 和 KDE。虽然有些用户对到底哪一种是真正的未来道路存有争议，但是两者都是相当完整的桌面管理器。实际上，您在使用其中的一种，并不意味着您不能使用另一种环境里的应用程序，此时只不过是一种不同的风格让您有一种在整个环境中的不连续感而已。

freedesktop.org 这个项目致力于建立一个环境，让应用程序能够与任何桌面环境都兼容。

22.5.1 KDE

KDE 代表 K Desktop Environment（K 桌面环境），它是用 C++语言编写，在 Qt 工具集库上构建的环境。喜欢养眼界面（如透明窗口、阴影和动画光标）的用户经常更愿意用它。它很漂亮，但在高端 PC 之外的地方运行比较慢。对于那些把大量时间用在到处点鼠标而不是运行应用程序的用户来说，在看和用之间的权衡最终决定了 KDE 是否为合适的选择。

从 Windows 或者 Mac 转过来的用户经常会因为 KDE 的漂亮图形而偏爱它。它也是热衷于全方位定制自己环境的用户最喜欢的桌面环境。对于其他人来说，KDE 用起来太啰嗦了，而 GNOME 是比较简单的选择。

为 KDE 编写的应用程序几乎都在名字里面有个字母 K，就像 Konqueror（Web/文件浏览器）、Konsole（终端模拟程序）或者 KWord（字处理器程序）。默认的窗口管理器 KWin 支持 freedesktop.org 的 Window Manager Specification（窗口管理器规范）标准，而且其外表可以配置，能够改变整体的风格，另外还有许多别的特性。KOffice 应用软件包括字处理、电子表格和演示工具。KDE 支持一套全面的开发工具，其中包括一个集成开发环境（IDE）。有了这些功能，KDE 提供了一个功能强大、风格统一的用户界面效果。

22.5.2 GNOME

GNOME 是另一种桌面环境，它用 C 语言编写，基于 GTK+ widget 库。它的下层对象通信模型使用了 CORBA（Common Object Request Broker Architecture，公用对象请求代理程序体系结构）。GNOME 这个名字源自于一个首字母缩写词，代表 GNU Network Object Model Environment（GNU 网络对象模型环境），但是现在这个起源已经不再适用了，GNOME 就只是一个名字而已。

GNOME 没有 KDE 那么花哨，也没有 KDE 配置性强，它的整体风格稍微有一点儿不统一。不过，它却特别整洁、快速和简单。大多数 Linux 发行版本使用 GNOME 作为其默认的桌面环境。

和 KDE 一样，GNOME 有一套丰富的应用软件。GNOME 应用程序通常可以从其名字里有个字母 G 辨别出来。标准的 GNOME 窗口管理器则是一个例外，它叫做 Metacity（发音像“opacity”），为实现风格可配而提供基本的窗口功能和外表。Metacity 遵守 GNOME 的模型，在设计上很简洁。如果您想要一些以前习惯的额外功能，像一个虚拟桌面或者灵巧地放置窗口，那么就需要像 **brightside** 或者 **devilspie** 这样的外部应用的支持（这正是 KDE 大显身手的一个领域）。

GNOME 上的 Office 软件包括用于字处理的 AbiWord，用于电子表格的 Gnumeric，以及 GNOME 孕育出的非常出色的项目之一 GIMP，它用于图像处理。GNOME 还带一个叫做 Nautilus 的文件管理器，以及 Web 浏览器 Epiphany。和 KDE 一样，GNOME 为应用软件开发人员提供了一个能扩展的基础设施。总而言之，GNOME 在一种易于使用的桌面环境里，为应用软件开发提供了一种强大的体系结构。

www.TopSage.com

22.5.3 KDE 和 GNOME 谁更好

在任何公共论坛上提这个问题，您都会看到“战火硝烟”是个什么样子。因为人们倾向于把桌面偏好转化成个人信仰，所以下面几段文字可能是本书中最不自以为是的话。

最好的答案是两种桌面都尝试一下，选择最能满足您自己需要的一种。牢记一点，您的朋友、您的用户和您的上司都对桌面环境有不同的偏好，这都很正常。

既然 freedesktop.org 正在建立统一桌面的标准，那么在 KDE 和 GNOME 阵营之间形成的相互敌视正在发展成一种为创造优秀软件而进行的健康向上的竞争。要谨记，您对桌面环境的选择并没有限制您可以运行哪些应用软件。无论您选择了哪一种桌面，您都能从这两种（以及其他）开放源代码项目开发出的完全互补的优秀软件中进行选择。

22.6 推荐读物

X.Org 的主页 x.org 包括了有关即将发布的版本的信息，还有到 X.Org 的维基、邮递列表和下载的链接。

Xserver 和 **Xorg** 的手册页介绍了普通的 X 服务器选项和 **Xorg** 所特有的命令行选项。它们都有对 X 服务器运行原理的全面概述。**xorg.conf** 文件的手册页介绍了这个配置文件，并且详细描述了它的各个配置段。该手册页还在 REFERENCES 部分里列出了显卡驱动程序。在此查找您的显卡，掌握驱动程序的名字，然后阅读驱动程序自己的手册页，就能了解到驱动程序特有的选项。

22.7 习题

E22.1 在网络上使用 SSH 运行一个程序 X。使用 **ssh-v** 核实正确地配置好了 X 转发功能。在您登录到系统里以后要把 DISPLAY 变量设为什么？运行 **xauth** 列出 cookie，核实已经启动了对那个显示的 magic cookie 身份验证功能。

E22.2 编写一个 shell 命令行或者脚本，分析 **xdpyinfo** 的输出，把当前屏幕的分辨率以 X×Y（例如 1024×768）的形式打印出来。

E22.3 研究 **Xorg** 的日志文件（**/var/log/Xorg.0.log**），尽可能多地判断出下面问题的答案。

a）出现的是什么类型的显卡，它使用哪个驱动程序？

b）这个显卡有多少显存？

c）用 EDID 探测该显示器的设置了吗？您是怎么知道的？

d）支持什么样的模式（分辨率）？

e）启用 DPMS 了吗？

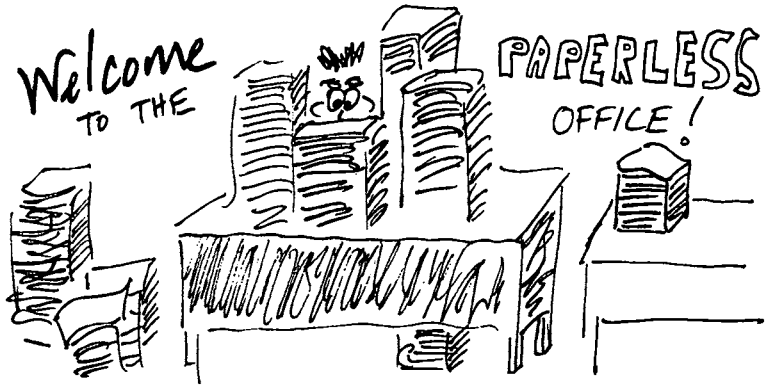
f）服务器认为物理的屏幕尺寸是多少？

g）鼠标的设备文件是什么？

E22.4 什么标志能禁止非本地的 TCP 连接连上服务器？说明这个选项为什么有用。

www.TopSage.com

第 23 章　打印



打印机的配置既烦人又困难。用户对打印的认识往往是想当然的，但是要让近在咫尺的打印机上呈现出完美的打印结果，要好好在管理工作上下一番功夫才行。

20 年以前，最常用的打印机是 ASCII 单行打印机。激光打印机过去不仅价格昂贵，而且十分稀有。高分辨率的输出设备需要有定制的驱动软件和定格式的程序。

现如今，激光打印机经常通过以太网或者无线链路连到 TCP/IP 网络，而不是通过串口或者并口连接到一台计算机。面对喷墨打印机，激光打印机已丧失了大部分低端市场。以前曾经是奢侈品的彩色打印机，已经像彩色照片和彩色显示器一样，变得司空见惯。找到一台黑白打印机很快就会变得和找到黑白胶卷一样困难。

在桌面和小型办公设备市场，专用打印机、扫描仪、复印机和传真机正在被能够完成上述所有工作的多功能设备所取代。有时候，这些设备甚至能读取数码相机存储卡里的文件。

正因为有了所有这些技术上的变化，您可能会指望 Linux 的打印系统极为灵活，而且实际上它们也的确如此。不过，这种灵活性是在最近才刚刚做到的。就在几年以前，大多数 Linux 的打印系统都还是以为古老的单行打印机开发的软件为基础。这些系统曾经为跟上不断演变的技术而历经修改、挖潜和重写，但却从未真正满足支持现代网络上的现代打印机的工作需要。幸好 CUPS（Common UNIX Printing System，公共 UNIX 打印系统）的到来解决了很多老系统上的缺点。

您可以在包括 Mac OS X 在内的大多数现代 UNIX 和 Linux 系统上找到 CUPS。虽然还有不多的几种打印系统仍在使用（比如 PDQ 和 LPRng），但是像 System V 的打印系统、Palladium、rlpr、PLP、GNUlpr 和 PPR 这样的早期打印系统都已经寿终正寝了。

在这一章里，我们集中介绍 CUPS 这种当前的事实标准。我们首先全面讨论打印机和打印术语，然后总体介绍 Linux 的打印系统，并概述 CUPS 的体系结构。接下来我们讨论打印机配置和管理的专门知识。最后，我们给出调试打印机的简要指导，浏览一下与打印相关的可选软件，还有一些通用的管理上的忠告，以此结束本章的内容。

www.TopSage.com

23.1 复杂的打印机

用户把打印机和其他外设，比如显示器和音箱放在一起考虑，但是这种看法没有给予打印机的复杂性应有的重视。从前，苹果公司制造的最强大的计算机叫做 Apple LaserWriter。现如今，您的台式机或许比您的打印机更强大，但是这台打印机仍然是一台计算机。它有 CPU、内存、操作系统，可能还有硬盘。如果它是网络打印机的话，它还有自己的 IP 地址和 TCP/IP 实现。

如果您有一台现代的网络打印机，那么就可以在您的 Web 浏览器中输入它的网络地址（例如，192.168.0.9）。这台打印机很可能会返回几个 Web 网页，让您管理这台打印机的硬件，这台打印机正在运行着自己的 Web 服务器。

既然系统管理员必须具有安全意识，那么您可能已经在想，“这意味着打印机可以受到拒绝服务攻击的入侵或者袭击吗？”您想对了。参见 23.7.8 节了解有关安全的内容。

您的打印机正在运行什么操作系统？什么？您不知道？别感到意外。不钻研一下的话，您可能也找不出来——或许钻研了也没搞清楚。操作系统随厂商的不同而不同，有时候甚至不同型号打印机的操作系统也不同。中档打印机和高端打印机甚至可能运行 UNIX 或 Linux 的某个变体系统[1]。

操作系统的混乱还只是个开始。打印机也能处理各种网络协议，接受以几种不同打印机专用的页面描述语言和文档描述语言说明的作业。

如果您正在管理一个比较大的设施，可能需要支持几家不同的制造商生产的不同型号的打印机。在您计算机上的打印软件必须做好准备，与不同的（有时候是未知的）硬件进行通信，并且使用一组通信协议。

23.2 打印机的语言

打印作业实际上是一个用专门编程语言编写的计算机程序。这些编程语言统称为页面描述语言（PDL，Page Description Language）。

用一种 PDL 编码的页面传送起来通常比等效的原始图像数据量更小、速度更快。用 PDL 描述还有与设备和分辨率无关的好处。

现如今最知名的 PDL 是 PostScript、PCL5、PCL6（也称为 PCL/XL 或者“pxl”）和 PDF。许多打印机能够接受多种语言的输入。我们在下面的几小节简要讨论一下每种语言。

打印机必须对这些语言写的作业进行解释，把它们转换为对实际的成像硬件有意义的某种位图表现形式。因此，打印机带有语言解释器。就像 C 或者 Java 一样，这些语言有多个版本，版本间也有不同。大多数 PostScript 打印机能够理解 PostScript Level 3，但是如果您把一个 Level 3 的程序发给了一台只懂 Level 2 的打印机，那么这台打印机可能会被搞糊涂。您会尝试用一个 FORTRAN 77 编译器去编译一个 FORTRAN 90 的程序吗？当然不会。

把 PDL 描述（或者其他任何东西，比如图像文件）由光栅处理器转换为位图形式的页面图像叫做“光栅图像处理”，能够执行光栅处理的程序叫做 RIP（Raster Image Processor，光栅图像处理器）。有时候把“rip”这个词不正式地用作一个动词。

在您的计算机上对打印作业做光栅处理，从显示器上看到图像也是可能的。我们将在 23.7.3 节讨论完成这个任务的基于主机的解释器，如 Ghostscript。理论上，您可以使用自己的计算机对打印作业

[1] 黑客们已经把 Linux 移植到 iPod 和 Xbox 上了，我们等着看谁先把 Linux 移植到 HP LaserJet 打印机上。

www.TopSage.com

做光栅处理，再把完成后（大得多）的位图交给一台不那么聪明的打印设备去打印。实际上，这正是许多 GDI（Windows）打印机的工作方式，Linux 也部分支持这种方式。

大家网 TopSage.com

23.2.1 PostScript

PostScript 是 Linux 系统上最常见的 PDL。它最初是由 Adobe 公司开发的，许多 PostScript 打印机仍然在使用从 Adobe 获得许可证的解释程序。几乎所有的页面布局程序都可生成 PostScript，有些只能用 PostScript。

PostScript 是一种完整的程序设计语言。您可以用文本编辑器或 **less** 来阅读 PostScript 程序。程序通常包括了大量的小括号、大括号和斜线，而且经常以字符%!PS 开头。虽然语言本身并没有要求在程序开头有上述形式的字符，但 PostScript 解释程序和其他打印软件在对打印作业进行识别和分类时通常需要查找这几个起始字符。

23.2.2 PCL

PostScript 的一种替代语言是惠普公司的打印机命令语言（Printer Command Language）。惠普打印机和许多别的打印机都懂这种语言，有些打印机只懂 PCL。和 PostScript 不同，PCL 只是告诉打印机如何打印页面，而前者是一种图灵完备的通用编程语言。PCL 作业是二进制的，常人无法直接阅读，通常要比等效的 PostScript 短得多。Linux 应用软件很少直接生成 PCL，但是过滤器能够把 PostScript 转换为 PCL。

和 PostScript 不同，PCL 的每个版本都只有一点儿不一样的地方。这种区别很小，但是却足以让人感到讨厌。在 LaserJet 5si 上正确打印的作业在 LaserJet 5500 上稍微会出点儿错，反之亦然。不只是这两种型号是这样，每种 PCL 打印机都有一种 PCL 方言，带有利用该打印机特性的自定义命令。

例如，如果您告诉计算机您有一台 LaserJet 4500，而您实际有一台 LaserJet 4550 的时候，就可能会生成一些 4550 忽略或者解释错误的 PCL 命令。如果您有一个保存下来的 PCL 打印作业——比如说，一张空白的采购需求表——而您把用于生成这个作业的打印机替换为某种更新型号的打印机，那么您就必须重新生成这个作业。

更糟糕的是，惠普制定了两种都叫做 PCL 但几乎完全没有关系的系列语言：PCL5（5C 表示彩色、5E 表示黑白）和 PCL6（也叫做 PCL/XL）。现如今，新的惠普打印机一般都有两种语言的解释器。

PCL4 是 PCL5 的老版本。对待 PCL4（或者更早）的打印机就像对待一个 Perl 4 解释器一样：用新版的替换它。

23.2.3 PDF

Adobe 公司的 PDF（Portable Document Format，可移植文档格式）由 Adobe Acrobat 软件和其他桌面出版工具生成。例如，OpenOffice 能够把文档导出为 PDF 文件。

PDF 是一种与平台无关的文档，一般用于针对在线和离线（打印好的）两种用途以电子方式交换文档。本书（英文版）的最后文本就是以一个 PDF 文件形式传给书刊打印机的。

PDF 是一种文档描述语言，它不只是一种页面描述语言。它不仅描述单个页面，也描述文档的整体结构：哪些页属于哪些章，哪些文本栏落入其他文本栏等。它还可以包含各种用于屏幕显示的多媒体功能。

有些打印机能直接解释 PDF。如果您的打印机不能，那么有一大堆 PDF 查看程序和翻译程序（包

www.TopSage.com

括 Ghostview、xpdf、kpdf、Evince 和 Acrobat Reader）可以把文档转换为其他（例如 PostScript）能被广泛理解的文档。您的打印系统甚至可以在把 PDF 文档发送给打印机之前，不用和您交互，就自动做文档转换。

23.2.4 XHTML

在这一系列语言中，XHTML-Print 才刚刚崭露头角。接收 XHTML-Print 数据流（例如，描述一个 Web 网页）的打印机会生成打印作业如实的呈现效果，但是不同的打印机可能产生不同的呈现效果，就像不同的浏览器对相同的 Web 网页显示方式不同一样。

用户为什么想这样？假如您是一位负责市场的副总，在您的手机上浏览 Web，要按您指定的呈现方式来看一个 Web 网页。您可以走到附近的有蓝牙功能的打印机旁边，从手机把 URL 发送给它。这台打印机完成其余的工作：它从 Web 下载那个网页、生成打印结果，并且打印若干页。您可以从打印机的输出纸匣拿到这些打印好的结果，用于您的报告。

23.2.5 PJL

PJL（惠普的打印机作业语言，Printer Job Language）不是一种真正的 PDL。它是一种描述打印机作业的元语言。我们之所以在这里介绍它，是因为您会看到在打印机的描述中提到过它。

PJL 是一种作业控制语言，它指定打印作业的 PDL、作业是双面还是单面，使用什么页面尺寸等这样的信息。PJL 命令出现在作业的开头部分，PJL 语句都以@PJL 开头：

```
@PJL SET COPIES=3
@PJL COMMENT FOO BAR MUMBLE
@PJL SET DUPLEX=ON
@PJL SET PAGEPROTECT=OFF
@PJL ENTER LANGUAGE=PCL
```

非惠普的打印机都广泛理解（或者刻意忽略）PJL，但是如果您在一台非惠普的打印机上打印带有 PJL 的资料时遇到问题，可以试着用一个文本编辑器删除 PJL 信息，重新打印这个作业。

23.2.6 打印机驱动程序及其对 PDL 的处理

把一个文件转换为某种特殊打印机所能理解的信息的软件叫做“打印机的驱动程序”。在 LaserJet 5500 上打印 PCL5，您需要 LaserJet 5500 PCL5 驱动程序。

如果一台打印机只支持您需要处理的语言的一个子集该怎么办？如果您从 Web 上下载了一个 PostScript 文件，而您的打印机只能懂 PCL 5E 怎么办？如果您的打印机不能直接解释 PDF，该怎样打印一个 PDF 文件？

一种选择是手工转换这个文件。Linux 主机都带有丰富的转换工具，几乎总是有办法把您有的东西转成您的打印机能打印的东西。浏览器可以把 HTML（或者 XHTML）转成 PostScript。OpenOffice 能够把 MS Word 文件转成 PDF。Ghostscript 能把 PDF 转为 PostScript，也能把 PostScript 转为包括 PCL 在内的几乎任何格式。

更容易的方法是让您的打印系统替您做这项工作。许多系统内部都知道需要做哪些转换，而且能为您自动设定这些转换。

如果您需要判断一个文件使用的是什么 PDL，而您不能从文件名（例如，**foo.pdf**）分辨出来，那么 **file** 命令可以告诉您（除非文件以一块 PJL 指令开头，在这种情况下，file 命令只会说“HP Printer Job

www.TopSage.com

Language data[惠普打印机作业语言数据]")。

把几个打印作业保存为文件，而不是把它们交给一台打印机，您可以看看用这些语言中的一种写出的程序是个什么样子。花一两分钟时间，在文本编辑器中读一下每种语言的文件，就能让您对它们的不同有个感觉。不要直接用 **cat** 命令把它们显示出来，因为只有 PostScript 是 ASCII 的。

PostScript：

```
%!PS-Adobe-3.0
%%BoundingBox: 0 0 612 792
%%Pages: 1
% ...
% Draw a line around the polygons...
pop pop pop dup 0 setgray 0 0 moveto dup 0 lineto 0.707106781 mul dup
    lineto closepath stroke
```

PDF：

```
%PDF-1.3
%Ã¢Ã£ÃÃ“
 81 0 obj
<<
/Linearized 1
/O 83
/H [ 915 494 ]
/T 125075
>>
endobj
 xref
81 24
0000000016 00000 n
 Ã^<8f>
^P^@Ã¤Ã‘<9e>
endstream
endobj
```

PCL5：

```
^[E^[&l1o0o1t0l6D^[&l1X^[*r0F^[*v0n1O^[*p4300X^[%1BDT~,1TR0TD1SP1FT10,50
CF3,1LB.~;^[%1A^[*c100G^[*v2T^[&a0P^[*p0X^[*p0Y^[(10U^[(s1p12vsb4148T^[&l0
E^[*p0Y^[*ct7920Y^[(10U^[(s1p12vsb4101T^[&a0P^[&l0o66f0E^[9^[&a0P^[*p0X^[*
p0Y^[*p474Y^[*p141X^[(10U^[(10U^[(s1p12vsb4101T^[*p402Y^[*p186X^[*v0O^[*c9
00a4b100g2P^[*v1O^[*p250Y^[*v0O^[*c900a4b100g2P^[*v1O^[*v0O^[*c4a156b100g2
P^[*v1O^[*p251Y^[*p187X^[*v0O^[*c899a154b10g2P^[*v1O^[*p346Y^[*p256X
```

PCL/XL：

```
Ã‘X^BX^BÃ¸<89>Ã^@Ã¸<86>Ã^CÃ¸<8f>AÃ^@Ã¸<88>Ã^AÃ¸<82>HÃ^@Ã¸(Ã^@Ã¸%Ã
Ã¸cÃ^Ã^P^@TimesNewRmnBdÃ¸Â¨Ã…UUÃ©BÃ¸Â¦Ãu^BÃ¸Âªo<85>Ã"Ã›^CA^BÃ¸Lk
Ãƒ^@^@Ã¸Â¡dÃ^Ã:^@
```

23.3 CUPS 的体系结构

CUPS 采用了一种客户机/服务器的体系结构。CUPS 服务器是假脱机程序，其中保存着客户机的

打印队列。

CUPS 客户机可以是有命令行接口（如 **lpr** 和 **lpq**）的应用软件，这些客户机也可以有图形用户界面（如 **kprinter**）。从某个服务器的角度来看，其他 CUPS 服务器也作为它的客户机。

考虑最简单的可能配置：在一台机器上有一个 CUPS 服务器，这台机器连接了一台打印机，而 CUPS 有一个打印队列。接下来的几小节简述几个常用打印任务中包含的命令和处理。

23.3.1 打印文档

下面的命令打印 **foo.pdf** 文件和**/tmp/testprint.pdf**：

```
$ lpr foo.pdf /tmp/testprint.ps
```

客户机把文件的副本传给 CUPS 服务器，服务器把它们保存在打印队列中。CUPS 在打印机准备就绪后依次处理每个文件。

CUPS 检查文档和打印机的 PPD（PostScript Printer Description，PostScript 打印机描述）文件，看看要正确打印文档需要做些什么（按我们后面说明的那样，PPD 甚至能用于非 PostScript 打印机）。

为了准备在一个特定打印机上打印一个作业，CUPS 把它传过一条由过滤器组成的管道线。这些过滤器能够执行各种功能。例如，一个过滤器能重新确定作业的格式，让实际的一页纸打印打出两页缩小的页面图像（双页并排，2-up），或者把作业从一种 PDL 转为另一种 PDL。过滤器还能执行特定于打印机的处理，如打印机初始化。一个过滤器甚至可以代替没有带自己的 RIP 的打印机执行光栅处理。

打印管道的最后阶段是一个后端，它把作业从主机通过一种合适的协议（如 USB）发送到打印机。这个后端也把通信状态信息返回给 CUPS 服务器。为了看到后端，试试命令：

```
$ locate backend | grep -i cups
```

CUPS 守护进程把打印作业传出之后，就回去操作它的队列，处理从客户机发来的请求。打印机则尝试打印交给它的作业。

23.3.2 查看和操控打印队列

lpq 命令从 CUPS 服务器请求作业的状态信息，为显示确定它的格式。

lpstat -t 报告打印服务器总体状态的汇总信息。

CUPS 客户机能要求服务器挂起、删除作业或者重新确定作业的优先级。它们还能把作业从一个队列移到另一个队列。大多数改动都要求用 **lpq** 报告的作业号来标识作业。例如，要删除一个打印作业，只要运行 **lprm** *jobid* 即可。

23.3.3 多台打印机

如果机器上连了不止一台打印机，那么 CUPS 给每台打印机都维护着一个打印队列。

命令行客户机接受一个指定打印机队列的选项（一般是**-P** *printer* 或者**-p** *printer*）。您也可以通过设定环境变量 PRINTER：

```
$ export PRINTER=printer_name
```

或者告诉 CUPS 针对您的账号使用一个特殊的默认值：

```
$ lpoptions -dprinter_name
```

www.TopSage.com

来设一个默认打印机。

在正常情况下，**lpoptions** 这个命令能设定您的个人默认打印机，默认值保存在**~/.lpoptions** 下。当以 root 身份运行该命令的时候，它设定**/etc/cups/lpoptions** 里的系统级默认打印机。**lpoptions -l** 列出了当前的选项。

23.3.4 打印机实例

如果您只有一台打印机，但是想以几种方式来使用它——比如说，既能用于快速草稿的打印，也能用于最终结果的正式打印——CUPS 能让您为这些不同的用途设置不同的“打印机实例”。

例如，如果您已经有一台名为 Phaser_6120 的打印机，下面的命令：

```
$ lpoptions -p Phaser_6120/2up -o number-up=2 -o job-sheets=standard
```

会创建一个名为 Phaser_6120/2up 的实例，执行双页并排打印（2-up）并增加标题页。命令：

```
$ lpr -P Phaser_6120/2up biglisting.ps
```

带标题页按双页并排方式打印 **biglisting.ps** 这个 PostScript 文件。

23.3.5 网络打印

从 CUPS 的角度来看，一个有许多机器的网络和一个独立的机器没有多大不同。每台计算机运行一个 CUPS 守护进程（**cupsd**），所有的 CUPS 守护进程彼此进行通信。

您通过编辑**/etc/cups/cupsd.conf** 文件配置一个 CUPS 守护进程接受从远程系统发来的打印作业（参见 23.4.3 节）。以这种方式设置的 CUPS 服务器，在默认情况下，每隔 30 秒就把自己对外提供服务的打印机广播一次。因此，本地网络上的计算机就自动了解到它们能用的打印机。

让打印机能让多个网络或者子网使用的话要多一点儿技巧，因为广播包不会穿过子网边界。常用的解决方法是在每个子网上指定一台从（slave）服务器，轮询其他子网的服务器获得信息，然后把这些信息转发给本地子网的计算机。

例如，假如打印机服务器 allie（192.168.1.5）和 jj（192.168.2.14）处于不同的子网，我们想让第三个子网（192.168.3）都能到达这两台服务器。为了达到这个目的，我们只要指定一台从（slave）服务器（比如说，copeland，192.168.3.10），把这几行加到它的 **cupsd.conf** 文件里：

```
BrowsePoll allie
BrowsePoll jj
BrowseRelay 127.0.0.1 192.168.3.255
```

前两行告诉从服务器的 **cupsd** 轮询 allie 和 jj 上的 **cupsd**，获得有关它们对外提供服务的打印机的信息。第三行告诉 copeland 把它了解到的全部信息转发给它自己的子网。

需要一个更复杂的设置？一台打印机多个队列，每个队列有不同的默认值？一台服务器通过把作业分摊给几个打印机实现负责均衡？多台服务器中的每一台都能互换处理同一种打印机的实例？LPD 或者 Windows 客户机？这里能详细介绍的变化太多了，但是 CUPS 能处理所有这些情形，CUPS 的文档可以告诉您具体细节。

23.3.6 CUPS 的支持协议：HTTP

HTTP 是 CUPS 服务器及其客户机之间所有交互的支持协议。CUPS 服务器监听 631 端口。客户

www.TopSage.com

大家网 TopSage.com

机用 HTTP POST 操作提交作业。状态请求通过 HTTP GET 操作实现。CUPS 的配置文件看上去也特别像 Apache 的配置文件。讲一点儿历史可以帮助您理解其中的原委。

最早的商业 UNIX 应用软件是文档处理程序。关键软件包括文本编辑器、标记语言（**nroff/troff**）和打印软件。

打印机相当原始，所以是假脱机程序。对于非 UNIX 系统也是这样，即便非 UNIX 系统是专有系统。IBM 系统知道如何驱动 IBM 打印机，Apple 计算机知道如何驱动 Apple 打印机，依此类推。您工作使用的计算机经常（正确地）被认为直连到了打印机上。对“串口还是并口？”这样问题的回答构成了打印机的配置。

有了网络打印机以后，问题出现了。早期的网络打印系统相当独特，打印机到假脱机程序的通信，客户机到假脱机程序的通信，以及网络流量协商上使用了各种各样的协议。

随着复杂性的增加，人们进行了几种尝试来创建统一的标准，但是没有一种得到广泛的认可。投入使用的协议也变得越来越旧、越来越不好。像双面打印这样的打印机新功能也促成了大量针对特殊情况的改进。

IETF 的打印机工作组咬紧牙关创立了基于 HTTP 的 IPP（Internet Printing Protocol，Internet 打印协议）。这个选择不仅用简单的 GET 和 POST 请求确定了交互通信的结构，而且还让打印操作能利用上广泛用于身份验证、访问控制和加密的标准技术。

ESP（Easy Software Products）的 Michael Sweet 和 Andrew Senft 把 IPP 以 CUPS 实现的形式引入 UNIX。现如今，CUPS 是世界上最完整的 IPP 实现。

虽然 ESP 有自己的市场定位和产品，但 CUPS 是一个开放源代码项目，可以自由地重新发布。现在的大多数 Linux 和 UNIX 系统都把 CUPS 作为其默认的打印系统。

一个 CUPS 服务器就是一个 Web 服务器，但这个服务器用 631 端口而不是 80 端口进行通信。您可以通过用一个 Web 浏览器连您的本地 CUPS 服务器（localhost:631）的方式来核实这个事实。您会看到，CUPS 服务器用一个 GUI 界面来提供其全部功能（您还可以在端口 433 上使用 SSL 实现和打印机的安全通信）。CUPS 使用 IPP 同 Web 浏览器、打印机、GUI 和 CLI 工具以及其他 CUPS 服务器通信。

23.3.7 PPD 文件

当您调用 **kprinter** 把 **book.ps** 打印到彩色打印机 Pollux 的时候，**kprinter** 可能会反过来问您想把它打印到多大的纸上。但请等一下——CUPS 如何告诉它的客户程序 **kprinter**，打印机 Pollux 能打印 A4 纸呢？CUPS 怎样知道 Pollux 能处理 PostScript 呢？如果 Pollux 不能处理 PostScript，CUPS 应该怎么办？CUPS 从哪儿得知 Pollux 是一台彩色打印机？

所有这些信息都保存在一个 PPD（PostScript Printer Description，PostScript 打印机描述）文件里，这个文件描述了一台 PostScript 打印机的属性和功能。CUPS 守护进程读取其打印机的 PPD，根据需要把有关打印机的信息传给客户机和过滤器。

PPD 一开始是为 Mac 开发的，但是 Windows 的软件很快就采用了它们。每种新打印机都带有厂商提供的一个 PPD 文件。Mac 和 Windows 打印机的驱动程序根据 PPD 文件得知如何把 PostScript 打印作业发送给打印机。例如，要求美国的一台单页黑白打印机在欧洲的 B4 纸上双面打印彩色文档则是没有意义的事情。

较老的 UNIX 和 Linux 打印系统不用 PPD。用户要么知道如何手工处理他们的 PostScript 文档，要么凑合用他们能得到的默认输出。与此相反，CUPS 的设计能彻底利用这个丰富的信息源。实际上，CUPS 非常依赖 PPD。

找到 PPD 文件要琢磨一下。如果 PPD 在您的机器上，那么可能在**/etc/cups/ppd** 或者**/usr/share/**

www.TopSage.com

cups/model 下。用 **locate .ppd** 命令可以把它们找出来。对于网络打印机而言，PPD 可能没有保存在本地，CUPS 的客户端要从相应的 CUPS 服务器获得 PPD 信息。

PPD 文件就是文本文件。看一下某个文件的内容，了解它所包含的信息类型可以提供不少资料。

PostScript 打印机都有厂商提供的 PPD，您可以从安装盘或者厂商的 Web 网站获得这些 PPD。CUPS 的发布库里带的 PPD 放在**/usr/share/cups/model** 下，CUPS 把当前在用的 PPD 复制到**/etc/cups/ppd** 下。

CUPS 也使用 PPD 来描述没有 PostScript 解释器的打印机。额外有个字段负责这个功能。

```
$ grep cupsFilter /usr/share/cups/model/pxlmono.ppd
*cupsFilter: "application/vnd.cups-postscript 0 pstopxl"
```

您可以用 **diff** 命令比较两个很接近的 PPD 文件（试试 **pxlmono.ppd** 和 **pxlcolor.ppd**），准确地看出两种打印机的区别。

如果您的打印机厂商没有提供 PPD 文件——可能因为该打印机没有 PostScript 解释器，厂商也不关心除 Windows 之外的其他操作系统——可以到 linuxprinting.org 通过 Foomatic 数据库搜索到更多信息。您的打印机也可能会得到 Gutenprint 项目的支持（gutenprint.sourceforge.net，以前的 Gimp-Print）。如果您从这些来源选了一个 PPD，您的用户想要获得尽善尽美的效果，那么就要试试每种选项，看看哪一种输出的结果最好。

如果哪儿都没有找到 PPD 文件，那么：

- 您应该在获得这种打印机之前先参考一下 linuxprinting.org；
- 可能有一种不错的通用 PPD 文件，即使它不能发挥您的打印机的全部功能，也会让您打印出点儿东西；
- 如果您对一个通用 PPD 文件进行了增强，使它用在您的打印机上效果更好，那么您应该把这个新 PPD 贡献给 Foomatic 数据库。

23.3.8 过滤器

CUPS 没有给每种打印机都使用一种专门的打印工具，而是使用了一系列过滤器来把一个要打印的文件转换为您的打印机能理解的东西。

CUPS 的过滤器方案相当不错。当您交给 CUPS 一个要打印的文件时，它就推断出这个文件的 MIME 类型、您的打印机能够理解的各种 MIME 类型，以及把前者转为后者所需的过滤器。

CUPS 使用**/etc/cups/mime.types** 下的规则判断出传给它的数据类型。例如，规则：

```
application/pdf       pdf string (0,%PDF)
```

的意思是“如果文件的扩展名为.pdf 或者文件内容以字符串%PDF 开头，那么这个文件的 MIME 类型就是 application/pdf”。

CUPS 通过在**/etc/cups/mime.convs** 文件中查询规则，判断出如何把一种数据类型转为另一种数据类型。例如：

```
application/pdf       application/postscript 33 pdftops
```

的意思是“运行过滤器 **pdftops** 可以把一个 application/pdf 文件转为一个 application/postscript 文件”。

如果您需要编写您自己的过滤器（不大可能出现这种情况），不要修改 CUPS 发布的文件。另外创建一组有自己基本名的文件，把这组文件放进 CUPS 能找到 PPD 文件的**/etc/cups** 目录下。CUPS 会读取所有名字后缀为**.types** 和**.convs** 的文件，而不只是 **mime.types** 和 **mime.convs**。

CUPS 管道线的最后一个环节是直接和打印机对话的过滤器。在非 PostScript 打印机的 PPD 中，

www.TopSage.com

您可以看到类似：

```
*cupsFilter: "application/vnd.cups-postscript 0 foomatic-rip"
```

或者甚至：

```
*cupsFilter: "application/vnd.cups-postscript foomatic-rip"
```

这样的行。

引号引起来的字符串和 **mime.convs** 文件中某一行的格式相同，但只有一种而不是两种 MIME 类型。这一行表明，**foomatic-rip** 过滤器把 application/vnd.cups-postscript 类型的数据转为打印机自身的数据格式。代价之所以为 0（或者省略），是因为做这一步只有一种途径，为什么要假装还有代价呢（Gutenprint 给非 PostScript 打印机的 PPD 略有不同）？

给定一个文档和一台目标打印机，CUPS 使用 **types** 文件确定文档的类型。然后它参考 PPD 判断出打印机要求什么样的数据类型，接着就用**.convs** 文件推出能把一种类型转为另一种类型的所有过滤器链，以及每条链的代价。最后，它选出代价最小的过滤器链，让文档通过那一系列过滤器进行处理。

在这条链上的最后一个过滤器把可以打印的格式传给后端，由这个后端通过任何一种打印机能理解的协议把数据发送给打印机。

试试 **locate pstops** 命令找到您的系统上可以使用的过滤器（**pstops** 是一种流行的过滤器，它能以各种不同的方式处理 PostScript 打印作业，如增加 PostScript 命令设定打印副本的数量）。其他过滤器也都在附近的位置。

您可以通过执行 **lpinfo -v** 命令，要求 CUPS 给出可以使用的后端列表。如果您的系统没有用于您所需要的网络协议的后端，那么可以从网上或者厂家那里得到它。

23.4 CUPS 服务器的管理

cupsd 在系统引导时启动，并连续运行。我们举例的所有 Linux 发行版本都默认以这种方式设置。

CUPS 的配置文件叫做 **cupsd.conf**，通常在**/etc/cups** 下可以找到它。这个文件的格式同 Apache 配置文件的格式类似。如果您了解两个文件中的一个，那么也就对另一个文件不陌生。

修改过这个配置文件的内容之后，要运行**/etc/init.d/cups** 重新启动这个守护进程，让所做的修改起作用（Debian 和 Ubuntu 这两个发行版本使用**/etc/init.d/cupsys** 重启它）。

默认的配置文件有很好的注释。文件里的注释加上 **cupsd.conf** 的手册页就足够了，我们这里就不再赘述这些内容了。

您可以手工编辑 CUPS 的配置文件，或者如果您安装有 KDE 桌面环境，那么可以通过 KDE Printer Manager 来配置该系统，从 KDE 的控制中心就能找到 KDE Printer Manager。

您可以用 **cupsdconf** 命令直接运行 KDE Print Manager 专门给 CUPS 的选项。这条命令包含在大多数系统的 **kdelibs** 这个软件包里，要用它没必要整个安装 KDE。

我们没有在生产环境中使用 KDE GUI 的经验，但是根据我们的测试，在我们援引的所有 Linux 系统上，它都报错说不能理解在默认的 **cupsd.conf** 文件中找到的某些选项。在 SUSE 上，它根本就拒绝执行，显然是因为 **cupsd.conf** 文件中的下面这行：

```
AuthType BasicDigest
```

让它去找根本不存在的文件**/etc/passwd.md5**（其他系统默认使用 `AuthType Basic`）。您遇到的情况可能有所不同。

www.TopSage.com

23.4.1 设置网络打印服务器

要让 CUPS 从网络上接受打印作业，需对 **cupsd.conf** 文件做两处修改。首先，把：

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
</Location>
```

改为：

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From netaddress
</Location>
```

然后，把 *netaddress* 替换为网络上的 IP 地址，从那个地址接受打印作业（例如，192.168.0.0）。然后找到关键字 `BrowseAddress`，将其设为网络的广播地址加 CUPS 端口的形式：`BrowseAddress 192.168.0.255:631`。

这两步告诉服务器接受从网络上任何主机发来的请求，并且向网络上每个 CUPS 守护进程广播它对外提供服务的打印机的信息。

要讲的就这么多！重启 CUPS 守护进程之后，它就作为服务器恢复运行。

23.4.2 自动配置打印机

实际上您没有打印机也能用 CUPS（例如，把文件从 PDF 转为传真格式），但是它的典型职能还是管理实际的打印机。我们在本小节考察您可以管理打印机的几种方式。

在有些情况下，添加一台打印机就是小菜一碟。在把 USB 打印机插到机器上的时候，CUPS 就会自动检测到它，并且判断出该如何处理它们。

打印机的制造商一般都提供安装软件，它能在 Windows 甚至 Mac OS X（该操作系统也使用 CUPS）上完成大多数配置工作。不过，您却不要认为厂商也会考虑如何在 Linux 上安装打印机。

即使您不得不亲自动手，但增加一台打印机经常不过就是把打印机插到机器硬件上、连上 localhost:631/admin 的 CUPS Web 界面，然后回答几个问题那么简单。KDE 和 GNOME 都带有自己的打印机配置窗口，您也可以喜欢用 CUPS 的界面。

如果别人加了一台打印机，网络上运行的一个或者多个 CUPS 服务器知道了它，您的 CUPS 服务器就会得知存在这么一台打印机。您不需要明确地把这台打印机加到本地的打印机目录中，也不用把 PPD 复制到您的机器。这都可以通过镜像做到。

23.4.3 配置网络打印机

网络打印机作为 TCP/IP 上的一个设备，需要它们自己的一些配置。特别地，它们需要知道自己的 IP 地址和掩码。这个信息通常用两种方法中的一种告诉它们。

首先，大多数现代打印机都支持通过网络从 BOOTP 或者 DHCP 服务器获得这些信息。这种方法

www.TopSage.com

在有许多同构打印机的环境中效果很好。参见 12.9 节了解有关 DHCP 的更多知识。

另一种方法是，您可以从打印机的控制台设一个静态 IP 地址，打印机前面板上的一组按钮和一个单行显示窗口就构成了打印机的控制台。浏览菜单，找到设 IP 地址的地方（如果有个菜单选项是打印菜单，那么可以用它把打印好的版本保存下来）。

有些打印机能让您通过一个串口访问虚拟的控制台。这个做法也挺好的，但是全部工作量可能和用前面板的按钮差不多。因为基本思路是一样的。

如果别的办法都不行，许多打印机都带使用手册。

网络打印机配好后，通常都有一个能从浏览器访问的“Web 控制台”。不过，在您能以这种方式访问它们之前，打印机先要有一个 IP 地址，所以要在引导的时候配好 IP。

打印机连上网而且能 ping 通之后，就要按 23.7.8 节的内容保护它的安全。

23.4.4 打印机的配置举例

让我们按下面的例子所示，从命令行添加并口打印机 groucho 和网络打印机 fezmo。

```
# lpadmin -p groucho -E -v parallel:/dev/lp0 -m pxlcolor.ppd
# lpadmin -p fezmo -E -v socket://192.168.0.12 -m laserjet.ppd
```

由此可见，groucho 连接到了**/dev/lp0** 端口，fezmo 的 IP 地址是 192.168.0.12。我们用 URI（universal resource indicator，通用资源指示符）的形式指定每个设备，从**/usr/share/cups/model** 下的文件里选一个 PPD。

只要把本地 cupsd 配置成了一个网络服务器，那么它立即就能让网络上的其他客户端用上新的打印机。

除了使用命令行接口之外，如果您喜欢，还可以使用 CUPS 服务器提供的基于 Web 的配置工具。对于本节介绍的管理任务来说，都能用 Web 配置工具。

CUPS 能接受的各种各样的打印机 URI。下面是几个例子：

```
ipp://zoe.canary.com/ipp
lpd://riley.canary.com/ps
serial://dev/ttyS0?baud=9600+parity=even+bits=7
socket://gillian.canary.com:9100
usb://XEROX/Phaser%206120?serial=YGG210547
```

有些 URI 带参数（例如，serial），别的 URI 则不带。命令 **lpinfo -v** 列出了您的系统能看到的设备和 CUPS 能理解的 URI 类型。

23.4.5 设置打印机的类

“类（class）”是共享一个队列的一组打印机。排在这个队列中的作业可以在任何一台先空出来的打印机上打印。下面的命令创建 haemer 类，其中包括 3 台打印机：riley、gilly 和 zoe。

```
# lpadmin -p riley -c haemer
# lpadmin -p gilly -c haemer
# lpadmin -p zoe -c haemer
```

注意，没有明确的一个步骤来创建这个类，只要给打印机指派了类，那么类就存在。实际上，CUPS 比这还聪明一些：如果网络上的多台打印机都有相同的名字，那么 CUPS 就把它们当作一个隐含的类，自动在这些打印机之间分担打印作业。

www.TopSage.com

大家网 TopSage.com

23.4.6 关闭服务

如果您想要删除一台打印机或者一个类，那么用 **lpadmin -x** 命令就可以轻松办到：

```
# lpadmin -x fezmo
# lpadmin -x haemer
```

但是如果您只想临时禁止一台打印机提供服务，而不是删除它，该怎么办呢？您可以在打印队列的任何一端阻塞队列。如果您禁用了队尾（队列的打印机一端或者输出一端），那么用户仍然能提交打印作业，但是提交的作业不会被打印。如果您禁用了队首（输入一端），那么已经在队列里的作业将被打印，而队列不允许再提交作业。

disable 和 **enable** 命令控制了队列的输出端，**reject** 和 **accept** 命令控制着作业的提交。例如：

```
# disable groucho
# reject corbet
```

该用哪个？接受在可预见的将来没希望打印出来的作业是个坏主意，所以使用 **reject** 命令延长不能打印的时间。对于不应该让用户发觉的短暂中断（例如，清除卡纸故障），可以使用 **disable**。

系统管理员偶尔会要个提示，帮助他们记住哪些命令控制队列的哪一端。考虑：如果 CUPS“拒绝（**reject**）”一个作业，意味着您不能“加入（**inject**）”它。另一种让命令好记的方法是，您能对打印作业做的是接受（**accept**）和拒绝（**reject**），而您能对打印机做的是禁用（**disable**）和启用（**enable**）。“接受（**accept**）”一台打印机或者一个打印队列是没有意义的。

警告：**enable** 除了是一条 CUPS 命令之外，它也是一条 **bash** 的内置命令。除非您指定了 **enable** 这条命令的完整路径**/usr/bin/enable**，否则 **bash** 就以为您的意思是指它自己的 **enable** 命令。**bash** 版的 **enable** 命令启用和禁用 bash 的内置命令，所以您可以用它禁用它自己[2]。

```
$ enable -n enable
```

CUPS 自身有时候会临时禁用一台它碰到问题的打印机（例如，如果有人拉掉了一条线）。一旦您修好了问题，要记得重新启用这个队列。如果您忘记了，**lpstat** 命令就可以告诉您。

23.4.7 其他配置工作

现在的打印机高度可配，CUPS 能让您通过它的 Web 界面以及 **lpadmin** 和 **lpoptions** 命令，调整各种各样的丰富特性。照理 **lpadmin** 用于配置系统级的任务，而 **lpoptions** 用于各个用户的任务。

lpadmin 命令能让您比 **disable** 和 **reject** 更细致地限制对打印机的访问。例如，您可以设置打印的额度，指定哪些用户能用哪些打印机。

23.4.8 纸型

在美国和加拿大，最常用的纸型叫做 letter，它的大小是 8.5×11 英寸。有些 Linux 发行版本（例如，Knoppix 和 SUSE）是欧洲出品，在欧洲甚至不知道英寸是什么，或者在英格兰，那里虽然知道英寸，但却不用英寸来量纸张大小。在这些地方以及在日本，常用的纸型叫做 A4，那里的打印机都带

[2] 在多长点儿见识，推断一下如何重新启用被禁止使用的 **bash** 内置的 **enable** 命令。**enable enable** 不起作用！

www.TopSage.com

A4 纸匣。因此，有些发行版本的打印工具默认生成的是 A4 的页面图像。

A4 纸之所以有意义，是因为它的不成比——数学上的不成比。A4 纸的长宽比为 $\sqrt{2}$。如果您把一张 A4 纸水平分成两半，那么就得到两个长宽比相同的半张纸。这个纸型叫做 A5。把 A5 纸对半就得到两张 A6 纸。从另一个方向看，A3 是 A4 面积的两倍，但是形状一样，依此类推。

换句话说，您可以造出 A0 纸，它的面积是 1 平方米，再用一把裁纸刀就能裁出您需要的其他纸型。能用来做相同游戏的唯一美国纸型是 ledger（11×17 英寸，也称为 tabloid），您可以把一张 ledger 纸对折，得到两张 letter 纸型的纸。

ISO 的 B 系列和 C 系列纸型也保留了 $1:\sqrt{2}$ 的比例，但是基准面积不同。B0 纸 1 米高，C0 纸的面积是 2¼平方米。工程师们会立即发现，B*n* 纸的边长是 A*n-1* 和 A*n* 边长的几何平均，而 C*n* 纸的边长是 A*n* 和 B*n* 边长的几何平均。

所有这些有什么含义呢？B*n* 和 A*n* 的样子一样，但是要比 A*n* 大，C*n* 居于两者中间。一份 A4 纸的报告正好能放在 C4 的纸盒里。把 A4 的信对折变成 A5 大小，正好放入 C5 的信封里。再对折正好放进 C6 的信封。

让情况稍微乱了一点儿的是，日本也有自己稍有不同的 B 系列纸型。虽然它的长宽比和 ISO 的纸一样，但是日本的 B4 纸型是 A3 和 A4 的算术平均，这让它比 ISO 的 B4 纸稍大一点儿。没有日本的 C 系列纸型。

正如 ISO 体系很容易把一本 B5 课本的两页内容复印到一张 B4 纸的讲义上一样，它也很容易做出各种 *n*-up（把几张缩小后的页面图像打印到一张纸上）打印效果。欧洲的复印机就有按 $\sqrt{2}$ 的比例放大或者缩小的按钮。

如果您的系统安装了 **paperconf** 命令，您可以用这条命令以英寸、厘米或者打印机的点（一英寸 72 点）为单位打印出各种名字的纸型的尺寸大小。对于美国人来说，表 23.1 列出了常见纸型的一些典型用途，让您对它们的大小有个感性认识。

表 23.1 ISO 纸型的常见用途

纸 型	常见用途	纸 型	常见用途
A0、A1	海报	A7	“3×5”索引卡片
A3、B4	报纸	B7	护照（甚至美国的护照都是 B7 的）
A4	通常意义的“一张纸”	A8	名片
A5	便笺（大约 5×8 英寸）	B8	扑克牌
B5、B6	书、明信片、德国的卫生纸		

遗憾的是，A4 纸比美国的 Letter 纸略微瘦了些，也略微长了些（8.3×11.7 英寸）。把 A4 文档打印到 letter 纸上一般会裁掉重要的两长条信息，如页眉、页脚和页码。反过来，如果您在欧洲或者日本，试着在 A4 纸上打印美国的页面，会把文档两侧的内容切掉一些（虽然问题不怎么严重）。

各个软件包有自己的默认纸型。例如，芬兰人 Markku Rossi 维护的 GNU **enscript** 默认用 A4 纸。如果您是美国人，而您的发行版本没有按不同的默认值来编译 **enscript**，一种选择是取得源代码重新编译。但一般更简单的做法是在命令行或者一个 GUI 配置文件中设置纸型。如果您的文档被裁掉了两端或者两边，那么纸型不匹配可能就是原因。

您还可以用 **paperconfig** 命令、环境变量 PAPERSIZE 或者**/etc/papersize** 文件的内容来为许多打印任务调整默认纸型（注意：**paperconfig** 和 **paperconf** 可不一样）。

23.4.9 兼容性命令

以前有两种互相竞争的打印系统：一种在 BSD UNIX 系统上，另一种在 System V UNIX 上。这两

www.TopSage.com

种系统的每一种都有相当简单的打印队列，而且提供创建、删除、启动、停止和暂停这个打印队列的命令，还提供了把单个作业排入队列或者从队列里取出的命令。

您可能会问，为什么会有两种系统，它们之间有什么显著的区别吗？站在 Linux 用户大会的中间喊一声“使用 **vi** 的人都是笨蛋！”——人们又会凑上来问我们为什么。

CUPS 很明智，它提供了替换这两种系统的兼容性命令。这样做的部分原因是方便习惯于以前系统的老前辈们使用，但是同现有软件兼容也是一个重要目标。

诚然，这些命令不一定和原来的功能完全一样，而且有些不常用的命令或者厂家特有的命令也没实现。许多使用这些命令的脚本也好好地用着 CUPS。把欠缺的地方当作一个机会：如果您想要为世界和平和帕累托最优（Pareto Optimality）贡献力量，仍然还给您留了要写的代码。

表 23.2 列出了 CUPS 带的 CLI 命令，把它们按照其来源分了类。

表 23.2　　CUPS 命令行工具及其来源

	命　令	功　能
CUPS	**lpinfo**	显示可用的设备或者驱动程序
	lpoptions	显示或者设置打印机参数和默认值
	lppassword	增加、修改或者删除摘要口令
	cupsdconf[a]	一个 CUPS 配置工具
	cups-config[a]	打印 cups API、编译器、目录和链接信息
System V	**lp**	打印文件
	cancel	取消作业
	accept, reject	接受或者拒绝提交作业给打印队列
	disable, enable	停止或者启动打印机和类
	lpstat	打印 CUPS 状态信息
	lpadmin	配置 CUPS 打印机和类
	lpmove	把一个作业移到一个新目的地
BSD	**lpr**	打印文件
	lprm	取消打印作业
	lpq	显示打印机的队列状态
	lpc	一个通用的打印机控制程序

a. 不要把这些相似的命令给搞混了。**cupsconf** 是 KDEPrint 中的一个 GUI 工具，**cups-config** 是 CUPS 包含的一个 CLI 工具。

23.4.10 常用的打印命令

除了把作业排入队列和打印作业之外，有关打印的命令还有很多。即使在一个普通的 Ubuntu 系统上，下面的命令：

```
$ man -k . | egrep -i 'ghostscript|cups|print(er|ing| *(job|queue|filter))'
```

也能列出 88 页以上和打印有关的手册页——那只是一个粗略的搜索结果而已（提起和打印有关的命令，需要说明一点，**print** 命令和打印一点儿关系也没有）。您应该了解一下这些命令和工具中的一些。

pr 是最古老的打印工具之一。它按照打印页的要求重新确定文本文件的格式。它把自己的输出定为每页 66 行，加上了页眉和页脚，并且文本为双倍行距。在把文本文件送到打印机前稍作处理上，它做得很完美。

Adobe 公司的 **enscript** 命令用一些花里胡哨的花样实现了类似的转换功能，它的输出也是 PostScript。GNU **enscript** 是该命令的一个开放源代码版本，它完全向后兼容 Adobe 的版本。不过 GNU

www.TopSage.com

enscript 中提供了丰富的新特性，这些特性包括对语言敏感的突出显示、支持各种页面大小、字体下载以及用户定义的页眉。

enscript 最知名的一点是它实现了双页并排（2-up）打印。如果您仍然因为这个特性而在使用它，那么可以试试 CUPS 给 **lpr** 的选项**-o number-up=2**。

按复杂性来排最复杂的是 Ghostscript，L. Peter Deutsch 最初编写这个软件是为了能在便宜的 PCL 打印机上打印 PostScript 文档。现如今，Ghostscript 既能解释 PostScript 也能解释 PDF。CUPS 把它当作一个过滤器来用，但是 Ghostscript 也能创建用于屏幕显示的页面图像，靠它自己，或者借助像 **gv**、GNOME Ghostview（**ggv**）或 KDE 的 KGhostView 这样的前端的帮助。

Linux 的发行版本都带一个自由版本的 Ghostscript，参考 www.ghostscript.com 了解更多信息。从 Artifex Software 公司可以获得有支持的商业版 Ghostscript。

23.4.11 CUPS 的文档

虽然 CUPS 的文档并不缺，但有时候您还是得去找它。诸如 lpr 的手册页这样的资料只是个大概。如果您没有在手册页中找到什么，那么不要以为您就找不到，在 google 上搜索吧。

安装 CUPS 的同时也带了很多 PDF 和 HTML 格式的手册。看这些文档有一个地方，连上 CUPS 服务器，点击在线帮助的链接。遗憾的是，如果您的问题是连不上 CUPS 服务器，那这就没什么用了。

同样的文档在 www.cups.org 也可以找到。在**/usr/share/doc/cups** 下也应该能找到。如果您的发行版本没有把文档装在那儿，那么试试：

```
$ locate doc | grep cups
```

另一种选择是查查发行版本的软件包管理器。

23.5 故障排查的技巧

在改了 **cupsd** 的配置文件之后，一定要记得重启 **cupsd**。您最好运行**/etc/init.d/cups restart** 重启（在 Debian 和 Ubuntu 上用**/etc/init.d/cupsys restart**）。您也可以通过 KDE Print Manager 应用重启这个守护进程。从理论上讲，您还可以给 **cupsd** 发一个 HUP 信号，但这似乎在 SUSE 系统上只会杀死该守护进程。

23.5.1 CUPS 的日志

CUPS 维护着三个日志：一个页日志、一个访问日志和一个出错日志。页日志是被打印的页的列表。其他两个日志和 Apache 的访问日志和出错日志类似。别感到意外，因为 CUPS 服务器就是一个 Web 服务器。

cupsd.conf 文件规定了记录日志的级别和日志文件的位置。它们一般都保存在**/var/log** 下。

下面是从一个日志文件截取的片段，该日志文件对应一个打印作业：

```
I [26/Jul/2006:18:59:08 -0600] Adding start banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Adding end banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Job 24 queued on 'Phaser_6120' by 'jsh'.
I [26/Jul/2006:18:59:08 -0600] Started filter /usr/libexec/cups/filter/pstops (PID
    19985) for job 24.
I [26/Jul/2006:18:59:08 -0600] Started backend /usr/libexec/cups/backend/usb
    (PID 19986) for job 24.
```

www.TopSage.com

23.5.2 直接打印的问题

要核实本地打印机的物理连接，可以直接运行打印机的后端软件。例如，当我们执行一台用 USB 连接的打印机的后端时得到的信息如下：

```
$ /usr/lib/cups/backend/usb
direct usb "Unknown" "USB Printer (usb)"
direct usb://XEROX/Phaser%206120?serial=YGG210547 "XEROX Phaser 6120"
    "Phaser 6120"
```

在 USB 连线不小心掉了（或者断了）的时候，这台打印机在后端的输出里的那一行就没有了：

```
$ /usr/lib/cups/backend/usb
direct usb "Unknown" "USB Printer (usb)"
```

23.5.3 网络打印的问题

在您开始追查一个网络打印方面的问题时，要先确保在实际连接打印机的那台主机上您可以打印。您的“网络打印问题”可能只是一个“打印问题”。同样，还要确保网络是好的。

接下来，尝试用一个 Web 浏览器（*hostname*:631），或者用 telnet 命令（**telnet** *hostname* 631）连到管理打印机的 **cupsd**。

如果您在调试网络打印机连接的时候遇到了问题，要牢记：在某台机器上必定有该作业的队列，有一种决定把作业发送到何处的方式，有一种把作业发送到有该打印队列的那台机器的方法。在打印机服务器上，必须有一个地方放作业队列，有足够的权限让作业被打印，而且有一条途径输出到打印设备上。

要追查这些问题，您必须检查几个地方：

- 在发出打印作业的机器上的日志文件，针对名字解析和权限问题；
- 打印服务器上的日志文件，针对权限问题；
- 在发出打印作业的机器上的 CUPS 日志文件，针对找不到过滤器、未知打印机、找不到目录等问题；
- 在打印服务器主机上的 CUPS 日志文件，针对错误的设备名、不正确的格式等消息。

系统日志文件的位置在**/etc/syslog.conf** 中指定。CUPS 日志文件的位置在**/etc/cups/cupsd.conf** 中指定。

23.5.4 发行版本特有的问题

CUPS 仍在演变，并且在不断发布对缺陷的修改。有些问题比其他别的问题更糟糕，而有些问题有安全隐患。在 Red Hat 的一些老版本上，CUPS 很不可靠。对那些系统的正确解决办法是升级操作系统。但是如果您不能安装一个新版的 Red Hat 或者 Fedora，那么就尝试安装最新版的 CUPS。

ESP（Easy Software Products）销售一种商业版的 CUPS，称为 ESP PrintPro，它支持的打印机类型要比自由版本的 CUPS 多得多。如果您必须支持一种不常见的打印机，在网上找不到必要的驱动程序，而 ESP 可能已经可以让它运行了。

www.TopSage.com

大家网 TopSage.com

23.6 实际使用打印机的问题

和打印机打交道会遇到麻烦，也会让人泄气。下面是一些基本的指导思想，能有助于约束那些问题的出现。当别的手段都不奏效的时候，只要乐观地想想您正在用的不是连到 RS-232 串口上的点阵打印机。当然，除非您用的真是这样的打印机。

23.6.1 打印机的选择

在您购买一款打印机或者接受别人正准备丢掉的一台“不要钱的”打印机时，要到 linuxprinting.org 的 Foomatic 数据库检查一下这台打印机在 Linux 下能得到多大程度的支持。该数据库把打印机分成从 Paperweight 到 Perfectly 的 4 类，您要 Perfectly 这类的。

CUPS 喜欢 PostScript 打印机。这些打印机的配置一般都很容易。非 PostScript 打印机也得到支持，但是没那么好。为了用这些打印机，您需要把打印作业转换为打印机最喜欢的 PDL 或者数据格式。这些软件要么可从 CUPS 发布里获得，要么可从本章介绍的其他某个地方获得。

23.6.2 GDI 打印机

Windows 仍然在两个方面具有优势，其中之一是它支持非常低档的打印机。Windows 系统上使用的廉价打印机统称为 GDI 打印机或者 WinPrinter。这些打印机几乎什么都不知道，没有对任何 PDL 的解释器。它们希望与它们相连的计算机做光栅处理。

和 GDI 打印机通信所需的一些信息，因为是用专门针对 Windows 的专有代码编写的，所以不公开。这样的保密措施阻碍了为这些设备开发 Linux 支持，但是开放源代码界在逆向工程上展现出了超凡的能力。CUPS 也能支持许多 WinPrinter 了。

Windows 有优势的第二个领域是它支持很新的打印机。新打印机和新显卡和新声卡一样，也是先发布 Windows 驱动程序，驱动程序支持该型号的设备所有公开和未公开的功能。Linux 的支持一般要滞后一段时间。如果您购买了一台刚刚发布的新打印机，因为需要它的一项高级功能，那么您可能不得不让自己先习惯一下，一段时间里只能从 Windows 系统驱动它。

23.6.3 双面打印

双面打印器（duplexer）是一个能让打印机在纸的两面都打印内容的硬件部件。有些打印机默认就带它们，而别的打印机把它们当作一种可选的配件来支持。

如果您没有（或者买不起）可以双面打印的打印机，那么可以先打一遍奇数页，然后把纸反过来再打一遍偶数页。用一份两页的文档实验一番，看看怎样把纸反过来，然后把使用方法标在打印机上。

许多打印软件都能帮助实现这个功能。例如，Ghostview（**gv**）有图标能让您标出奇数页或者偶数页的任何一组，还能指定一个选项只打印标出来的页。**lp** 和 **lpr** 的 CUPS 版本用选项**-o page-set=odd** 和**-o page-set=even** 处理这个任务。如果您经常使用这些功能，您可以让一个“打印机实例”记住这些选项，参见 23.3.4 节。

有些打印机，特别是便宜的激光打印机，设计的时候没有考虑过双面打印。它们的制造商经常警告说在一页纸的两面打印会造成无法修复的损伤。我们实际上从未看到过这样的损伤，但是打印机制造商肯定也不会误导您。他们会吗？

www.TopSage.com

23.6.4 其他打印机配件

除了双面打印器之外，许多打印机也能让您增加内存、额外的纸匣、硬盘和其他配件。这些升级工作要么能让打印作业难以消化，要么能让打印效率更高。如果您在打印作业上遇到问题，要检查出错日志，看看打印机有更多内存的话是否有助于解决问题。参见23.5.1节。

23.6.5 串口和并口打印机

如果您的打印机直接用一条线连到了您的计算机上，那么它在用某种形式的串行或者并行连接。

虽然并口的标准还未完全过时，但它的确给我们提供了几乎不要配置的端口。如果您有一台并口打印机，那么可能很容易配置——只要您的计算机也有一个并口。

Mac机器上的串行连接是FireWire（火线），但是Linux上的串行连接一般用USB。查看www.qbik.ch/usb/devices或者www.linux-usb.org上给出的Linux支持的USB设备数据库，了解您的硬件目前获得支持的情况。

您几乎用不着老式的RS-232串行打印机了。如果您确实碰到了这样的打印机，它要求大量额外的配置。假脱机打印软件必须知道波特率和其他串口参数的正确值，使之能够和打印机正确地通信。您可以在该设备的URI里指定所有这些参数。比起推断出使打印机工作所需的各种串行参数组合，可能购买一种不同的打印机速度更快。

23.6.6 网络打印机

许多打印机带有完整的网络接口，这可以让它们直接接到网络上，并且通过一种或多种网络或打印协议接受打印作业。数据发送到连网打印机的速度要比发送到串口或并口上的打印机快得多。

23.7 给打印机的其他建议

和打印有关的一些管理问题超出了Linux和CUPS的细节。这些问题大部分的原因在于，打印机是一种每次用的时候都要花钱，但又反复无常的机械设备。

23.7.1 只在必要时才用标题页

CUPS能够给每个作业打印标题页和结尾页，显示出这个作业的标题和提交它的用户信息。对于由许多不同的人使用的打印机来说，这些标题页有时候能帮助把作业区分开，但是在大多数情况下，它们都是对时间、墨粉和纸张的一种浪费。

我们建议您在CUPS的GUI上（或者通过运行**lpadmin**）彻底关闭标题页，然后对个别使用标题页可能有好处的作业才把这个功能打开：

```
$ lpr -o job-sheets=confidential gilly.ps
```

您还可以通过使用**lpoptions**为个别用户打开标题页功能。另一种可以考虑的选择是采用一个给作业增加标题页的打印机实例（参见23.3.4节）。

如果需要，您可以复制**/usr/share/cups/banne**下一个现有的标题页，对其进行修改，创建自定义

www.TopSage.com

的标题页。给这个新的标题页起个新名字，和其他标题页放在一起。

23.7.2 提供回收站

所有种类的计算机用纸都是可回收的。您可以把纸张原来的包装盒当作回收箱；纸正好能放进去。在它上面贴上标签，告诉人们不要把异物（比如订书针、曲别针和报纸）丢在里面。

23.7.3 使用预览程序

用户经常会打印文档，发现格式上的一个小错，修改好了之后，又要重新打印整个文档。采用能让用户在屏幕上预览打印结果的软件，就能够轻而易举地避免这种对纸张和时间的浪费。

光有预览软件还不够，您的用户还得知道如何使用它们。他们通常都是碰巧学会的。记账记录的用途之一就是检查出相同的文档被重复打印的情况。有时候通过这个记录就能知道哪些用户不知道预览软件。

许多现代的 WYSIWYG（所见即所得）编辑器、浏览器和打印作业合并软件里都内置预览功能。对于其他类型的文档，您的选择不是一成不变的。Ghostview（**gv**）这样的工具能预览任何 PostScript 和 PDF 文件。对于 **roff**，要把 **groff** 的输出用管道送入 Ghostview；对于 TeX 的输出，尝试 **xdvi**、**kdvi** 或者 Evince。

23.7.4 购买便宜的打印机

打印机硬件技术是一项成熟的技术。您不必花很多钱就能买到有高质量输出且结构可靠的打印机。

除非确实需要，否则不必购买价格偏贵的“工作组”打印机。如果您只打印文本，那么一台便宜的“个人”打印机就能提供高质量的输出，而且几乎同样快速而可靠，更不必说轻上好几十磅了。输出速度为每分钟 10 页的打印机可同时服务 5 位全时的录入人员。给一个有 25 位录入员的小组购买 5 台价格为 250 美元的打印机要比购买一台价格为 1250 美元的打印机更实惠。

一般来说，决不要从计算机制造商那里购买打印机（或者硬盘、内存）。他们的打印机通常是贴牌（译者注：就是把其他厂商的产品拿来换上自己的商标）的产品，但价格却是原来打印机的两倍。用户最好还是购买用于 PC 及 Macintosh 市场的 PostScript 打印机（有些公司，像惠普既制造计算机也制造打印机。它们都不错）。

即使您坚持采用主流品牌，但也没有哪个制造商能保证万无一失。我们已经有惠普激光打印机的丰富使用经验。它们都是很耐用的产品，惠普也一直在积极地支持 Linux 和 CUPS。即使如此，有些惠普的打印机却是彻头彻尾的灾难。在购买之前先看看 Internet 上的评论。

此时便宜也是一个优点：250 美元的错误要比 1250 美元的错误更容易挽回。

23.7.5 手头多备墨盒

激光打印机偶尔需要更换它们的墨盒。在打印出的页面上出现发淡或者空白的地方暗示出打印机的墨粉快用完了。在您需要更换墨盒之前就先买好准备着。在紧急情况下，可以把墨盒从打印机中取出，轻轻摇一摇它，把剩下的墨粉给摇匀。这样您经常可以让一个墨盒再多打几百页。

出现纹路和斑点可能意味着您应该清洁一下打印机。看看打印机上是否有一个“clean（清洁）”功能。如果没有，或者清洁功能也没有帮助，请仔细阅读制造商的清洁指南，或者花钱对打印机做

www.TopSage.com

保养。

打印机制造商讨厌循环使用和退出市场的墨盒，他们竭尽全力地试图阻止这样的墨盒。许多设备使用了“加锁”的耗材，打印机通过电子的或者物理的方式检测这些耗材的身份。即使两台打印机看上去一模一样（如 Xerox Phaser 6120 和 Konica-Minolta Magicolor 2450），也并不意味着您可以在这两种打印机上使用相同的墨盒。

有时候您可以做做手术，把一家厂商的墨盒改造成另一家厂商的打印机能用的墨盒，但知其所以然才有帮助。通常您只会搞得一团糟。如果您把墨粉撒了出来，那么要用吸尘器把尽可能多的墨粉吸掉，再用冷水把剩下的擦干净。与人们的常识相反，虽然激光打印机的墨粉不会像所有的细粉末一样，对健康或者环境造成危害，但最好还是避免吸入墨粉粉尘。

换好墨盒之后，把新墨盒的包装箱和包装袋留下来，在循环使用旧的墨盒时再用。然后查查电话簿或者 Web，找一家公司接手旧墨盒。

加锁耗材刺激了仅以新墨盒几分之一的价格给旧墨盒重新装粉的公司的成长（“打孔灌粉”）。墨盒的回收公司经常也是“打孔灌粉”公司，所以您可以同时循环使用旧墨盒，又准备替换墨盒。

对循环使用的墨盒在质量上和在寿命上的看法也不一致。我们知道的一家“打孔灌粉”公司不重新灌彩色墨粉，也不销售二次加工货，因为他们认为节省的费用要比使用它们而增加的维护费少得多。

23.7.6 注意单页成本

打印机制造商使用 MBA 称之为“计量（metering）”的概念，产品的全部成本与客户对产品的使用量尽可能呈线性关系。这就是为什么墨粉和墨水过于昂贵，而新的打印机硬件有时候却要按低于制造成本的价格销售的原因。

在写本书（英文版）的时候，一家制造商买的彩色激光打印机价格为 299 美元。而更换它的一套完整墨盒要花 278 美元。您可以从沃尔玛以不到 50 美元的价格买到一台喷墨打印机，但是很快您就会花比打印机还要贵的价格购买一套更换用的喷墨墨盒。

您可以对此发发牢骚，但是打印机公司总得从什么东西上给自己赚钱吧。墨盒越便宜意味着打印机更贵。一个原则是，只要您不用喷墨打印机打印东西，那它们就便宜，激光打印机一开始成本高，但是耗材更便宜，而且用的时间更长。

喷墨打印机打一页全彩色的内容，成本是激光打印机打类似一页的 20～50 倍。它还要用专门的纸，打印速度更慢。喷墨墨盒空得很快，要频繁更换或者报废。墨水遇水就化——不要用喷墨打印机打印厨房里用的菜谱。另一方面，您现在可以用喷墨打印机打照片，结果看上去就像是从暗房冲洗出来的一样。有彩色激光打印机吗？还没那么好。

所有的打印机都是机械易损件。便宜的打印机坏得更快。

换句话说，它就是所有因素的折中。对于用量不多的个人用途——一天打印一两张 Web 网页，或者一个月打一卷胶卷——那么低成本的通用喷墨打印机就是极好的选择。

下次您去买打印机的时候，在您买打印机之前，要估计一下打印机您要用多长时间、您的打印量有多少以及用它做哪种打印。量化估计每种备选打印机每页纸的长期成本。并且问问您的“打孔灌粉”公司，他们是否二次加工这种打印机的墨盒，价钱是多少。

23.7.7 考虑打印机记账

对于中到大型基础设施，即使您并不打算收打印机的使用费，也要考虑对打印机的使用记账。给每个打印作业记账的开销并不重要，而您却能得知谁正在使用打印机。打印机记账还能提供有关打印作业各种来源的信息，这些信息对于规划新打印机的部署来说是有价值的。

www.TopSage.com

人们已经为 CUPS 开发出了几种打印机记账软件（如 **accsnmp** 和 PyKota）。

23.7.8 保护打印机的安全

网络打印机一般都支持远程管理。您可以从网上通过 IPP 或者 SNMP，或者用 HTTP 从 Web 浏览器来配置和监视它们。通过远程接口，您就能设置打印机的 IP 地址、默认网关、syslog 服务器、SNMP 群体名、协议选项以及系统管理口令这样的参数。

在默认情况下，大多数可以远程管理的打印机都没有得到保护，必须在安装过程中指定一个口令给它（也可能是一个 SNMP 群体名）。请参考您的打印机厂商为您提供的安装手册，学习如何在特定的打印机上完成此项任务，不过 CUPS 和 KDE Print Manager 的 GUI 管理工具逐渐可以把厂商间的差异掩盖起来。希望这个趋势能延续下去。

23.8 KDE 的打印功能

我们在本章已经提过几次 KDE[3]。KDE 的打印工具确实相当华丽，不过应该对它们多做一些说明。KDE 投入了很多力量来让它的打印工具和接口与下层的打印系统无关。在 CUPS 流行起来以后才有它，所以它能发挥 CUPS 的所有功能。但是它也能配合从 LPRng 到通用外部程序的每样东西。

GNOME 的打印工具比 KDE 的要落后，但是 GNOME 的开发人员也想要用户得到良好的打印体验。开发工作进展迅速，等您读到本书的时候，GNOME 的打印功能就可以和 KDE 的相媲美了。本章早期草稿的一位读者曾经这样说这个颇具讽刺意味的事实，CUPS 取代了那些“不死不活却势不两立的打印标准”，只不过是为桌面打印工具套件之间的竞争铺平了道路。

KDEPrint 是在 KDE 下重要的打印框架。KDEPrint 提供了增加打印机、管理打印任务、重启打印服务器等工具。没错，CUPS 也能让您做到这些。KDEPrint 工具存在的理由有两个。

首先，它们有 KDE 的外观和风格，对 KDE 用户保持了一致。例如，**kghostview** 工具把 Ghostview 封装在一个更像是 KDE 的界面下面（您可能已经注意到甚至 KDE 工具的名字都有一种独特的风格）。有人最近问我们 **ksh** 是不是个 KDE 应用软件。

其次，KDEPrint 和假脱机程序无关。如果出于某个原因，您不能运行 CUPS（或者在更糟的情况下，您必须在打印系统间来回切换），您仍然可以使用 KDEPrint 来管理打印工作。要先说明一点，CUPS 比其他打印系统功能更强，所以如果您退到另一种打印系统上，那么可能有一些 KDEPrint 的功能就没有了。

如果您在 shell 里完成打印工作，为什么应该为所有这些 GUI 界面而操心呢？因为您的用户可能没有在用 shell 接口，所以您最终不得不为了支持 KDE 界面而对它有所了解。

下面是您应该对 KDEPrint 有所了解的主要地方。

- **kprinter**，一个提交打印作业的 GUI 工具。
- Add Printer 向导，自动检测网络打印机（JetDirect、IPP 和 SMB）和一些本地连接的打印机。Add Printer 向导也能让您添加和配置它没有自动检测到的打印机。
- Print Job Viewer，移动和取消打印作业，显示打印作业的状态信息。
- *KDEPrint Handbook*，系统的文档。通过 KDE Help Center（KDE 帮助中心）能够获得该文档，

[3] KDE 是一组库和用户接口的标准，用于 X 窗口系统下运行的图形界面，Linux 的所有 GUI 均以 X 窗口系统为基础。另一种选择方案是 GNOME 系统，它是大多数发行版本的默认系统。除了界面外观不同，实际上没必要在 KDE 和 GNOME 之间做出选择。参见 22.5 节了解 GNOME 和 KDE 的总体介绍。

www.TopSage.com

但很难找到，这很烦人。比较容易的一种途径是调用类似 **kprinter** 这样的应用，然后点击 Help。另一种方法是运行 **konqueror help:/kdeprint**。KDEPrint 文档的另一个来源是 printing.kde.org。

- Print Manager，打印系统的主 GUI 管理工具。它也有点儿难找。虽然随发行版本的不同，它在菜单层次结构下的位置不同，但您可以在桌面主菜单中到处找找。另一种选择是运行 **kcmshell printmgr** 或者 **konqueror print:/manager**。

Add Printer 向导和 Print Job Manager 既可以通过 **kprinter**，也可以通过 KDE Printer Manager 调用（不要在 Konqueror 中用 URL **print:/manager** 和 **print:/printer**）。

KDEPrint 的每个用户的信息都保存在**~/.kde** 下。用户可以直接阅读这个文件，但是应该通过 Print Manager 来修改。调整它您要冒风险。

23.8.1 kprinter：打印文档

kprinter 是 **lpr** 的 GUI 替代软件。它可以从命令行以相似的方式调用。您甚至可以不显示 GUI。

```
$ kprinter --nodialog -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

等价于：

```
$ lpr -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

您的用户可能想用 GUI。给他们演示如何把文件从文件管理器或者桌面拖到 **kprinter** 对话框里，然后批量打印所有的文件。在他们的浏览器的打印对话框里，用 **kprinter** 代替 **lpr**，这样一来，他们就有了一个 GUI 风格的打印对话框。教他们用鼠标点中“Keep this dialog open after printing（打印后保持对话框为打开状态）”复选框，他们每次想要打印的时候，甚至都不用等着重新启动这个程序。

注意“Print system currently in use（当前使用的打印系统）”菜单，表明了 KDEPrint 对打印系统的中立态度。还要说明一点，**kprinter** 也在没有实际打印机的情况下提供了“打印到 PDF”和“打印到传真”功能。这些高级选项值得关注，您可以把您的简历放入打印队列，并且指出在您的老板回家以后再打印。

23.8.2 Konqueror 和打印

许多 Web 浏览器都认识一组专门的 URI，它们起到了调用独特功能的通道作用。您可能至少在 Firebox 里试过 about:config 和 about:mozilla。类似地，print:系列的 URI 是 Konqueror 到 KDEprint 的通道。

print:/这个 URI 向您展示出了所有的可能。print:/jobs 监视打印作业，print:/manager 从 Konqueror 里启动 Print Manager。

注意，您这里没有和 CUPS 打交道，至少没有直接打交道。这都是 KDEPrint 层的组成部分。

23.9 推荐读物

Sweet, Michael. *CUPS: Common UNIX Printing System*. Indianapolis, Indiana: Sams Publishing, 2001.

我们在本章里已经提到过几次 linuxprinting.org。它是 Linux 打印资源的大宝库，答疑的良好起点。这个站点也有一个不错的 CUPS 教程，其中包含一个介绍故障排查的部分。

您可以在 cups.org/newsgroups.php 找到与 CUPS 有关的一大堆新闻组。这是个答疑的好地方，但是应先做好您自己的功课，再有礼貌地提问。

www.TopSage.com

KDE 有 KDEPrint 命令的手册页，也带 *KDEPrint Handbook*。您可以在 printing.kde.org 找到更多信息。所有这些资源都包含对其他文档的援引，很有用（即使您没有 KDE，KDE 的文档也包含了有关 CUPS 的很多通用资料）。

23.10 习题

E23.1 使用 Web 浏览器访问您网络上的 CUPS 服务器。是什么不让您对该服务器的打印机做管理性的修改？

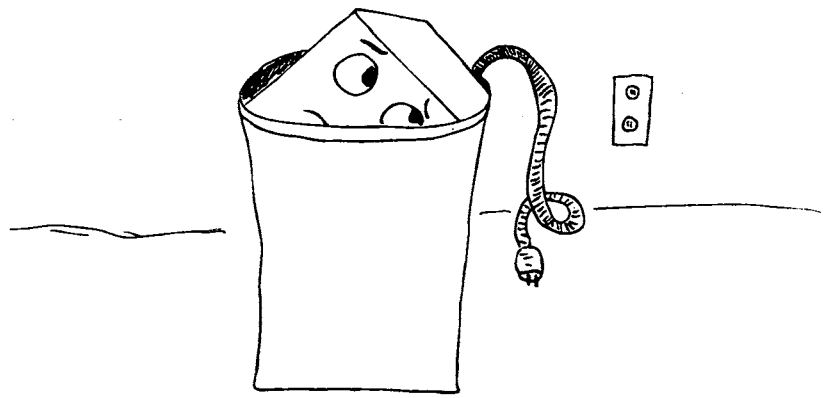
E23.2 找到某个计算机盲（一个文科生、您的母亲或者可能是一个微软认证专家），教这个人在 Linux 系统上打印一份 PDF 文档。您发现有让人糊涂的任何步骤吗？您怎样能使这个过程对别的用户更容易？

E23.3 访问一个真实的或者虚拟的硬件商店，如 Sam's Club 或者 Amazon.com，列出 200 美元以下您能购买的打印机。如果您必须明天给自己的单位购买其中的一种打印机，那么应该买哪一种？为什么？采用从 linuxprinting.org 数据库获得的数据来支持您的分析。

E23.4 要求您为基于 Linux 的激光打印机设计系统软件，瞄准企业工作组市场。您从哪个 Linux 发行版本开始？您会把其他哪些软件包括进来，您必须编写什么软件？您如何接受 Windows 和 Mac OS 的客户机？（提示：查出专用于“嵌入式系统”的 Linux 发行版本。）

www.TopSage.com

第 24 章　维护与环境



随着桌面工作站的兴起，依靠大型主机的计算平台则逐步销声匿迹，中央机房（也叫做“数据中心”）余下的日子曾经一度都能掰着手指头数出来。别担心！过去 10 年中，桌面系统愈发依赖运行 Linux 这类操作系统的中央服务器。因此，成群的服务器现在又在那些一度被遗弃的机房里热闹起来。

为服务器提供良好而且易于维护的环境又和从前一样重要了。实际上，最新的 1U 机架式服务器[1]对供电和空调的要求往往和它们所取代的大型机相当，或者更高。

本章主要提供有关处理和维护硬件以及向其提供良好环境的一些提示。

24.1　维护基础

硬件维护在传统上由费用昂贵的年度维护合同来规定。虽然仍然可以轻易获得这样的合同，但如今采用“自力更生”的方法维护硬件则更常见也更划算。

如果坚持使用日志，那么浏览一下过去 6～12 个月内的记录，您就能对故障率有个概念。由此可见，保持对设备故障及零部件更换情况的详细记录可以使用户据此准确地评估可采用的维护方案。有些部件出现故障的频率比制造商预想的要高，因此维护合同有时候具有既方便又经济的优点。但要记住，总有一天所有的硬件都应该更换，而不要再维护了。了解您的硬件，在它最终寿终正寝的那天让它平和地离开吧。用户可以考虑将这些过于陈旧的配件捐助给当地的大学或学校。对于这些学校来说，再老的设备也会有用处。参考 30.6.3 节了解有关硬件报废的更多知识。

在规划您的维护策略时，要考虑到哪些是最有可能提前老化的设备部件。设备里如果包括经常动的部件，那么就要比固定不动的设备（比如 CPU 和内存）可靠性低得多。下面是一些系统管理老手们通常关注的对象：

[1] 一“U”的高度是 1.75 英寸，这是测量机架空间的标准单位。

www.TopSage.com

- 磁带机；
- 自动装带机和换带机；
- 硬盘；
- 风扇；
- 键盘；
- 鼠标；
- CRT 监视器。

24.2 维护合同

有几家大公司对非本公司制造的计算机设备也提供维护服务。可以说，这几家供应商经常热衷于取代原来的制造商，于是就插上一脚。用户有时能通过让生产厂商与第三方提供商展开竞争的办法谈成极富吸引力的维护合同。如果可能的话，最好能从您认识或信任的人那里搜集到可能提供维护服务的所有商家的参考信息。

24.2.1 现场维护

如果用户有现场维护的合同，则技术服务人员将直接为设备带来备件。承保的响应时间从 4 小时到 24 小时不等，一般会在合同中规定。响应时间在上班时间内会比一周里的其他时间更短。

如果您正在考虑一份快速响应维护合同，通常值得计算一下，保持有两套完整的备份系统，可以通过交换来替换掉发生故障的计算机，这样做的成本有多少。针对如今硬件价格很低的现状，比起哪怕是最奢侈的维护合同来，交换整个系统往往能做到维修得更快，而且这样的投资经常是最少的。

24.2.2 板卡更换维护

板卡更换服务要求您和您的员工来诊断问题，也许还会得到来自制造商服务网点的热线员工的帮助。诊断结束后用户需要拨打维修电话，说清问题，并且定购所需的替换板卡。通常制造商会立即发货，而用户则可在第二天收到。接下来要做的就是安装板卡，让硬件恢复，然后将替换下来的板卡装回新板卡的包装盒。

制造商一般都会给购销交易分配一个“RMA（return merchandise authorization，返厂认证）”编号。在送回已损坏的板卡时，应确保在发货单据上填写该编号。

24.2.3 保修

制造商的产品保修期的长短应该对计算该设备在使用期内的维护费影响很大。在大多数情况下，最好的保修方案或许要算“选择性承保”了。硬盘制造商提供长达 5 年的承保，而有些内存模块甚至能达到终身承保。对于计算机来说，标准保修期为一年，但几年或者更长时间的保修期也并非鲜见。在购买新设备的时候，货比几家获得最好的承保——长远来看将会节省您的资金。

在许多单位里，为固定资产设备似乎要比为人员或维护更容易获得投资。我们偶尔才会为新硬件的“延伸保修”（这又称为预付费的维护）而付钱，从而把设备费转换为维护费。

对于包含了许多部件的硬件来说，在系统安装起来不久很快就会出现维护和可靠性方面最大的问题。通常把系统安装好以后一、两天内出现的硬件故障称为“早期致命故障（infant mortality）”。

www.TopSage.com

24.3 处理电路板的知识

电路板和其他电子设备应当注意轻拿轻放，不要将咖啡溅在这些电路板上，也不要把书堆放在电路板上等。大多数客户工程师（伴随维护合同而来的那些和蔼的维修人员）对电路板的处理比理应的处理要粗暴 10 倍。

24.3.1 静电

电子部件对静电都十分敏感。为了安全地处理板卡，必须在安装前以及安装过程中让您自己保持接地。将地线的一端缠在手腕上，而将另一端系在“接地”（通常可以用三相插座的第三个插头），这样才能正确地保护到您。

请记住，当您第一次打开一个电子部件的包装时，或者在处理电子部件的任何时候——不仅仅在最后安装的时候，都需要注意静电。您收到邮包（以及您禁不住要打开这个包裹）时如果所在的办公室铺有地毯的话要格外小心，地毯比硬地板产生的静电更厉害。

减少铺地毯的地板上的静电的一种方法是，从超市买一个喷壶，用一份织物软化剂兑 30 份水构成溶液。每月一次把该溶液喷在地毯上（注意不要洒到计算设备上），这样使其静电保持在较低的水平。上述步骤还可以保持办公区域的气味清新。

24.3.2 重插板卡

您偶尔会通过切断电源、擦干净接口卡（SCSI 卡、以太网卡等）边缘的触点、重新插卡，再打开电源恢复系统的方法来解决硬件问题。如果经过这种处理可暂时解决问题，但在一个星期或一个月后同样的问题又出现了，那么板卡和主板之间可能电气接触不良。

触点可以用一种特殊的清洁方法和清洁工具擦干净，或者用普通的铅笔擦擦干净。注意不要使用老化变硬的橡皮。如果橡皮不能很好地擦掉纸上的铅笔字，那它也不能很好地对触点进行清洁。尽量不要用手触摸这些触点。用铅笔擦（一种柔软的研磨料）“擦”触点并刷掉留下来的橡皮屑，然后再重新安装板卡。

24.4 监视器

过去几年里，我们已经高兴地看到，LCD（液晶）显示器的价格已经降到能普及的水平了。虽然 LCD 显示器的初始成本要比 CRT 显示器略高，但是这类设备耗电低，维护费用低，并且比以前的 CRT 显示器不易造成视觉疲劳。如果您仍然在单位里使用 CRT 显示器，那么简单地把它们更换为 LCD 显示器就是一项不错的维护计划。

如果您仍然得维护 CRT 显示器，那么要注意，它们中有许多显示器的亮度和对比度只能从电路板进行调节。遗憾的是，CRT 显示器内部往往使用上万伏的高压并且在断电以后还能保持相当长的时间。因此，为了防止触电，我们建议一定要由有资格的技术人员进行调整。用户不要自己去尝试。

24.5 内存模块

目前大多数硬件使用的内存均为 SIMM（Single Inline Memory Modules）类型、DIMM 类型（Dual

www.TopSage.com

大家网 TopSage.com

Inline Memory Modules）或 RIMM 类型（Rambus Inline Memory Modules）而不是单个芯片。这些模块的容量从 32MB 到 4GB 不等，所有的模块都位于一块小小的电路板上。

如果需要为工作站或者服务器添加内存，一般要从第三方供货商订货并自行安装。不要从工作站供应商处购买内存，他们会向您敲竹杠[2]。在添加内存时考虑安装大容量的。尽管内存价格不断下跌，但一块普通主板上的标准扩展槽数也越来越少了。

在定购内存之前，一定要反复检查您的系统文档，确保您已经搞清楚了系统能接受的内存类型。安装支持更高总线速度或者特殊功能的内存，比如 DDR（Double Data Rate），常常能够提高性能。要掌握每个系统上有多少内存插槽，在增加新的内存模块时是否有什么限制。有些系统要求成对增加内存模块，而别的系统则不严格要求这样，但是在内存模块成对时能发挥更高的性能。

要了解新老内存模块彼此是否能一块用。在大多数情况下，只有所有的模块有相同的特性或者速度才能在实际中一起用。升级的时候有时值得去掉系统原来的内存。

如果需要自行安装内存，则要牢记内存比其他任何设备对静电更敏感。在打开一整包内存的之前您要确保有良好的接地。

同样可以采用本章前述的方法，经常用铅笔擦对内存模块进行清洁。

24.6 预防性维护

听上去似乎显得有点原始（我们中有些人认为我们已经犯不上为此费事了），不过许多硬件中的确带有空气过滤器，这些过滤器需要定期进行清洗或者更换。阻塞了的过滤器会阻止空气的流通，从而可能会导致过热，这是硬件故障的一种主要原因。保证所有设备上的通风孔畅通不被阻塞是十分重要的，但要对那些紧紧装在 1U 或者 1U 机箱里的服务器特别注意。这些设备要靠自身的通风装置来降温。不通风的话，肯定要出问题。

有活动部件的任何设备都应注意定期润滑、清洁，并对传动带进行维护。用户可根据这些老式设备工作时所发出的声响判断其工作状态并对其做相应的维护。

在服务器系统中，最容易出故障的部分是风扇和电源模块——尤其是在 PC 上。PC 上的这一部分经常是单独的一个现场可更换部件（field-replaceable unit，FRU）。请定期检查服务器以确保其风扇转得又快又有力。如果不是如此，通常必须更换整个电源部件——不然的话，您可能就在冒使设备过热的风险。不要试图对风扇本身进行润滑，这一方法可能会推迟一定会发生的故障，但它也能加速问题的出现或导致对系统其他部件的损坏。

许多 PC 的机箱都有能安装第二个风扇的位置（以及给它供电的电源插座），如果不考虑噪声的话，安装第二个风扇总是可取的做法[3]。除了降低计算机部件的运转温度之外，如果主风扇发生故障，那么多加的风扇还可以充当备份风扇。多加的风扇很便宜，要有两个作为后备。

处于多尘环境中的计算机烧毁部件的情况会比处于相当清洁环境中的计算机频繁。灰尘可以阻塞过滤器、使润滑变干、卡住活动部件（风扇），同时元件表面覆盖一层“绝缘的”灰尘还将大大影响其散热的能力。所有这些因素都会提高系统运转时的温度。用户应定期检查并清洁过滤器。如果系统所处的环境比较恶劣，则还需不定期对系统内部进行清洁（铺有地毯的房间可能算恶劣的）。

真空吸尘是最好的除尘方法，但要保持吸尘器的马达距离系统部件和磁盘至少 5 英尺远，使电磁干扰降至最低。应对机房定期吸尘，但这项工作最好交由训练有素的专业人员来完成，他们知道保持正确的距离，不会损坏设备（办公室的门卫通常不是完成此项工作的合适人选）。

[2] 如果内存是作为大宗采购一部分的话，则另当别论，有些这样的采购还是很不错的。

[3] 或者了解一下最新的超静音风扇。

www.TopSage.com

磁带机通常也要求进行定期清洁。大多数盒式磁带机都是插入特殊清洗磁带进行清洁的。

24.7 环境

计算机和人一样，如果它能在一个十分良好的环境中工作，就能工作得更好，工作时间也更长。尽管它们不太会对窗外的景色计较多少，但您确实需要关注它们所处的环境的其他方面。

24.7.1 温度

计算机系统理想的运行温度是 64℉～68℉之间（即 17℃～20℃之间），同时湿度约为 45%。遗憾的是，这一温度与计算机用户的理想工作温度并不一致。机房达到 80℉（27℃）以上的室温则暗示出计算机内部的温度约有 120℉（49℃）。目前商业级芯片的运行温度上限为 120℉，到达此温度时它们将停止工作，而超过 160℉（71℃）时它们将会损坏。入口温度至关重要，一台机器的散热口一定不要对着另一台机器的入风口。

24.7.2 湿度

大多数计算机硬件所要求的理想湿度范围是 40%～55%。如果湿度太低，静电则会成为一个问题。如果湿度太高，则可能在电路板上凝结出水滴并由此导致短路和氧化。

24.7.3 办公室降温

现在，许多计算机都待在人们的办公室里，它们必须在建筑物内的空调下生活（通常这些空调在晚上和周末被关闭），而且要克服放在散热通风口上的大量报纸和书籍的影响。当将一台计算机放入办公室时，要记得它将会偷用原本为人服务的空调。HVAC（Heating、Ventilation and Air Condition，采暖、通风和空调）工程师对放有计算机的办公室的实际制冷负载的估计出奇地糟糕。如果您是能够影响到对制冷容量估算的人，那么有一条好规则，房间中每个人产生的热值为 300 BTUH，而每台计算机平均将产生 1100 BTUH 的热量。不要让工程师们忘记为任何接受阳光直射的窗户增加太阳造成的制冷负载。

24.7.4 机房降温

如果您足够“走运”，把您的服务器机群搬进了那种具有活动地板的机房中的一个，此类机房一般建于 20 世纪 80 年代，具有足以给您的所有设备降温的制冷容量，那么首先应该考虑的最大问题是尽快找到能够对这种原始的冷却系统进行维护的专业人员。对我们中间其余的人来说，选择容量正确的制冷系统是长期以后产生差别的原因。总之，降温良好的机房将是一个幸福的机房。

我们发现，复查 HVAC 工程师提供的冷却负荷估算数据，尤其是在为机房安装系统的时候，确实是一个好主意。您肯定需要一位 HVAC 工程师来帮助您分别对影响环境的屋顶、墙壁及窗户（太阳的辐射热不能忽略）进行制冷负载的估算。HVAC 工程师通常对这些因素有着丰富的经验，而且应该能提供给您一个准确的估计。您要核查的部分是机房内部的热负载。

您将需要确定以下因素的热负载：

www.TopSage.com

- 屋顶、墙壁及窗户（见 HAVC 工程师所提供的估算数据）；
- 电子设备；
- 照明设备；
- 操作人员。

一、电子设备

您可以根据服务器（及其他电子设备）所消耗的功率来估算热负载，直接测量耗电量是迄今获取此信息的最好方法。附近友好的电气工程师也经常能帮助您做这件事情，或者您可以购买一台廉价的仪表亲自上阵[4]。除此方法之外，大多数设备都以瓦为单位标识出其所能消耗的最大功率（虽然典型的功耗要比这个最大值小）。您可以将这个数据换算成标准热量单位 BTUH 的数值（乘以 3.412 BTUH/瓦）。例如，如果要在所建机房内安装 25 台服务器，每台服务器的功率均为 450 瓦，则计算结果：

（25 台服务器）•（450 瓦/服务器）•（3.142 BTUH/瓦） =38 385 BTUH

二、照明设备

与电子设备类似，同样也可根据照明设备所消耗的功率估算其相应的热负载。典型的办公室照明系统包括 4 个 40 瓦的荧光灯管。如果您的新机房装有 6 只这样的灯管，则计算结果将是：

（6 个照明设备）•（160 瓦/照明设备）•（3.412 BTUH/瓦）=3 276 BTUH

三、操作人员

有时需要有人进入机房维护某些设备。为每人计入 300 BTUH 的热负荷。如果所设计的机房希望同时允许进入 4 人：

（4 人）•（300 BTUH/人） =1 200 BTUH

四、总的热负载

在计算好每个部分的热负载以后，将它们加起来即可得到总的热负载。对本例来说，假定 HVAC 工程师对屋顶、墙壁及窗户的热负载的估计值为 20 000 BTUH。

20 000 BTUH（屋顶、墙壁及窗户的热负载）
38 385 BTUH（服务器及其他电子设备的热负载）
3 276 BTUH（照明设备的热负载）
1 200 BTUH（操作人员的热负载）

62 861 BTUH（总的热负载）

制冷系统的容量通常以 ton 为单位。您可以通过除以 12 000 BTUH/ton 将 BTUH 转换为 ton。您还应该加上至少 50%的富余因素作为误差和将来的增长：

（62 681 BTUH）•（1 ton/12 000 BTUH）•（1.5） = 7.86 ton（所需冷却负载）

看看您的估计与 HVAC 工程师提供的估算值能有多大程度的一致。

24.7.5 温度监视

如果您所支持的是一个重大的计算环境，那么甚至在您不在的时候也能监视机房温度（及其他环境因素，例如噪声和电源）是个不错的主意。星期一早上在机房的地上看到一堆已融化的塑料是件非常令人失望的事情。幸运的是，自动机房监视器在您离开的时候也能监视一切。我们使用并推荐的是 Phonetic Sensaphone 的系列产品。这些便宜的产品可以监视诸如温度、噪声及电源等环境变量并在发现问题时以电话（或你的传呼机）方式提示用户。

[4] P3 出品的“KILL A WATT”表是一种普遍选择，价格大约在 30 美元。

www.TopSage.com

24.8 电源

计算机硬件要求配备轻巧、稳定而且清洁的电源。在机房中，这意味着要有一台昂贵的电源稳压器，它可以滤除瞬间高压并能调节输出适当的电压和相位。在办公室中，位于计算机和墙壁之间的电涌保护器也将有助于把硬件与电源瞬间高压隔绝。

用户可能会考虑到把服务器和网络基础设备接到一个 UPS（Uninterruptible Power Supply，不间断电源）上。好的 UPS 具有 RS-232、以太网或者 USB 接口，能与被供电的设备相连。这条连接能让 UPS 警告计算机，说电源已经出现故障，计算机应该在电池耗尽之前稳妥地关闭它自己。

一项研究估计，全美耗电量的 13%用于计算机操作。传统上，UNIX 系统的基础是期望一天 24 小时不间断供电的软硬件。现在只有服务器和一些网络设备确实需要一直开机。对于台式机来说，如果用户有方便的关闭它们的方法，则应在晚上切断电源（如果您信任用户能正确做到的话）。参考 2.6 节了解关机过程的有关信息。

您偶尔会发现自己置身于这样一种情形之下，即因为一个内核或者硬件上的问题而不得不由您定期重启一台服务器。或许您在机房里还有一台非 Linux 的服务器，更容易碰到这类问题。不论上述哪一种情况，您可能都想安装一个系统，让您通过远程控制来重启有问题的服务器。

APC（American Power Conversion）公司推出了一种不错的解决方案。该公司的产品 MasterSwitch 类似于一个配电板，所不同的是该设备能通过其内置的以太网口用 Web 浏览器对它进行控制。

24.9 机架

现在，带有活动地板的机房——电源、空调、网络连接和电话线都藏到了地板下面——已经过时了。你是否曾经在活动地板下的迷宫里试着找出一条电缆来呢？我们的经验是，这种“经典的”活动地板尽管透过玻璃看上去很美，但无异于“老鼠洞”。现在，您应该用活动地板来隐藏供电线路、分散制冷风扇，此外别的什么也不要装。

如果您的目标是以专业的方式运行计算设备，那么用于服务器级机器的专用机房则是必须的。服务器用机房不仅能够为您的机器提供舒适、可控温的环境，而且还可以满足设备在物理安全方面的要求。

在专用机房中，将设备放入机架（与放在桌上或地上正相反）是唯一可维护的、专业的选择。如今，最佳的设备放置方案是使用通过架空布线系统相互连接的机架。这种方法带来了令人着迷的高技术感，同时也不会牺牲组织性或可维护性。

最好的悬轨布线系统是由 Chatworth Products 公司制造的。使用标准的 19 英寸单轨电信机架，您就可以为导轨式服务器和机架式服务器构筑它们的安乐窝。利用两个 19 英寸背对背式的电信机架可构建有高技术感的“传统”机架（适用于不需要在设备的前后附加其他机架硬件的情况）。Chatworth 公司的产品包括机架、电缆布线槽、电缆处理工具以及把机架安装到建筑物中所需的其他全部硬件。由于机架中的电缆都布于线槽中，因此十分容易跟踪它们，您会自发地去保持它们的清洁。

24.10 数据中心标准

服务器机房已经变得非常普及，很多组织都开始为它们建立标准。这些标准一般规定了内部网络

www.TopSage.com

连接种类、冷却和电源供给（以及这些资源的后备方案）以及每年的设备停运维护时间。Uptime Institute公布了一组这样的标准，它们按类型总结在表 24.1 中。

表 24.1 Uptime Institute 服务器标准

级 别	连续工作时间	电源/冷却	冗 余 性
I	99.671%	单路	没有冗余部件
II	99.741%	单路	有部分冗余部件
III	99.982%	多路，1 路活动	具有冗余部件，同时维护
IV	99.995%	多路，>1 路活动	具有冗余部件，完全故障容错

Uptime Institute 除了深入描述每一级别的标准以及如何达到标准之外，还就故障容错数据中心的基础设施，对各种专题提供了统计和最佳实施方案的信息。您可以在 www.upsite.com 网站看到这些信息。

24.11 工具

对系统管理员来说拥有一套完备的管理工具将会大大提高其工作效率。配备专用工具箱对于能在最短的时间内对系统的突发事件做出处理来说是非常重要的。表 24.2 中列出了您可能应该在工具箱中包括的一些工具，或者至少应该放在能随手拿到的地方。

表 24.2 系统管理员的工具箱

普 通 工 具	
十字头螺丝刀：0 号、1 号和 2 号	镊子
一字头螺丝刀：1/8 英寸、3/16 英寸和 5/16 英寸	剪刀
电工刀或者瑞士军刀	套筒扳手套件
尖嘴钳和普通钳	小号手电筒或者笔形手电筒
小号螺丝刀	六角扳手套件
4 盎司圆头锤	八角扳手套件
计算机专用工具	
剥线钳（带有剪线刀刃）	便携式网络分析仪
线缆扎带（和 Velcro 公司的类似产品）	备用的电源线
备用的 5 类 RJ-45 交叉线	RJ-45 接头压钳
备用的 RJ-45 接头（硬芯、多股）	SCSI 终结器
数字万用表（DMM）	
静电接地带	
杂 项	
紧急维修联系人名单[a]	棉花棒
随叫随到的支持人员的家庭电话和呼机号	绝缘胶布
急救工具箱	牙科镜
6 扎上好的啤酒[b]	手机

a. 如果可行的话，还应加上维护合同。
b. 建议的最小值。

www.TopSage.com

TopSage.com

24.12 推荐读物

下面这些资料提供了有关数据中心标准的更多信息。

Telecommunications Infrastructure Standard for Data Centers. ANSI/TIA/EIA 942.

ASHRAE INC. *ASHRAE Thermal Guidelines for Data Processing Environments*. Atlanta, GA: ASHRAE, Inc., 2004.

EUBANK, HUSTON, JOEL SWISHER, CAMERON BURNS, JEN SEAL, AND BEN EMERSON. *Design Recommendations for High Performance Data Centers. Snowmass*, CO: Rocky Mountain Institute, 2003.

24.13 习题

E24.1 为什么想把计算机安装到机架上？

★E24.2 环境因素既会影响人，也会影响机器。把本书所列的因素和您自己的一些因素（例如，灰尘、噪声、强光、杂乱等）综合起来。选出4个因素，评估您的机房对于人和机器的适宜性。

★E24.3 一台工作站用电0.8A，它的监视器用电0.7A（在120V的电压下）。

a）这个系统耗电多少瓦？（提示：P=EI）

b）电价是大约0.10美元/千瓦时，让这个系统运行一整年要花多少钱？

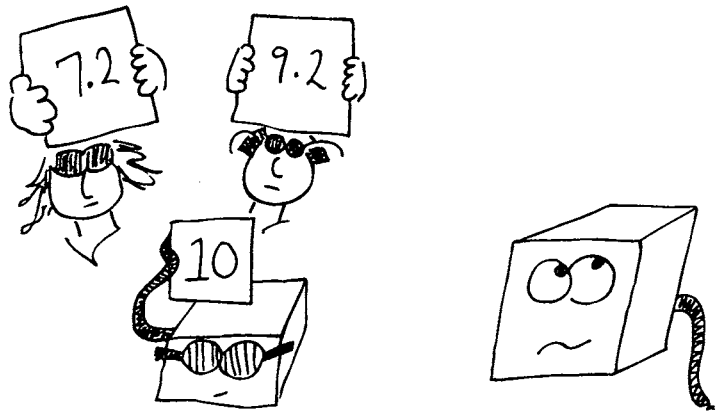
c）每天平均关机16小时（要么手工关、要么使用“显示器电源管理”这样的“能源之星”功能）的话，每年节省多少钱？

d）冷却这个系统每年要花多少钱？（说明您对冷却成本所做的假设，给出计算过程。）

★★E24.4 为您的站点设计一个新机房。说明在有关空间、机器数量以及每台机器的类型和电源负载方面所做的假定。然后计算这个机房所需的电源功率和制冷要求。要把服务器和客户端工作站都算进去。还要包括房间的布局、照明和预计的人数。

www.TopSage.com

第 25 章　性能分析



本章集中讨论作为服务器使用的系统的性能。台式系统遇到的性能问题一般不会与服务器所遇到的类型相同，对于如何提高桌面机器性能这样的问题，答案几乎总是“升级硬件”。用户喜欢这样的答案，因为这意味着他们能更频繁地让自己用上好的新桌面系统。

Linux 有别于其他主流操作系统的地方之一就是，它提供的有关其内部工作情况的数据量更多。系统的每个层面都有详细的信息，系统管理员控制着各种各样能够调整的参数。如果您在判断性能问题的原因上遇到了麻烦，那么总是可以看到源代码的。出于这些原因，Linux 经常是注重性能的消费者所选择的操作系统。

遗憾的是，Linux 的性能调优并非易事。用户和系统管理员都经常认为，如果他们知道正确的“妙方”，那么他们系统的速度就会加倍。人们有的一种常见的梦想就是调整控制着调页系统和缓冲池的内核变量。现在 Linux 主流发行版本的内核大都经过预先调节以保证系统在各种负载状态下都具有合理的（虽然确实如此，但不是最优的）使用性能。如果您试图基于某种特定的性能指标（例如，缓冲利用率）对系统进行优化，则会增加扰乱整个系统相对于其他性能指标和负载条件表现的机会。

最严重的性能问题往往要归咎于应用软件，很少和下层的操作系统有关系。除非您在单位内部开发这样的应用，否则您要么是运气不佳，要么就得花上大把时间向应用软件商的支持人员电话咨询。本章讨论系统级的性能调优，而把应用级的调优留给其他章节。

总而言之，您应该对在 Web 上读到的任何东西都持保留态度。在系统性能这一领域内，您会看到，在所有主题上都有表面上挺有说服力的观点。但是，这些理论的大多数支持者并不具备设计验证试验所要求的知识、训练和时间。大众的支持意味着没有支持，对于每种浮躁的提议，您都会看到如同古希腊歌舞剧合唱诗般的赞美：“我已经按 Joe 所说把缓存扩大了 10 倍，我的系统感觉起来非常非常快!!!”情况就是这样。

www.TopSage.com

性能调优是一项艰巨的工作，要求有耐心、有系统性的方法，并且要做仔细的分析。您应该谨记的基本规则有以下几点。

- 收集并检查系统的历史性信息。如果系统一个星期前还性能良好，那么检查最近对系统进行更改的方面可能会引导您找到元凶。始终在裤子口袋中装有常规基准资料以备不时之需。
 第 19 章讨论的一些趋势分析工具也适用于性能监视。25.3.6 节介绍的 sar 也能充当穷人的趋势分析工具。
- 调整系统的方法一定要让您和以前的基础进行比较。
- 调整实事上让情况更糟的时候，一定要有退回从前的计划。
- 不要有意使系统或者网络过载。Linux 会给每个进程造成它可以有无限资源的错觉。但一旦使用了 100%的系统资源，Linux 就不得不吃力地工作来维护那一假象，这会拖延进程的执行，而且往往这么做的本身就消耗了很大一部分资源。

25.1 做什么可以提高性能

这是为提高系统性能肯定能做的几件事情。

- 确保系统具有足够的内存。正如我们在后面将会看到的那样，内存大小对系统性能有着主要的影响。目前内存的价格是如此便宜，因此您完全能为每台重视性能的机器购买足够的内存。
- 复查系统和各个应用的配置。许多应用都能以多种方式进行调优，获得很大的性能提升（例如，把数据分布到多块磁盘上、不要随时执行 DNS 查询或者让受欢迎的服务程序运行在多台实际的机器上）。
- 改正使用上的问题，包括由用户引起的问题（同时运行过多的作业、低效的编程习惯、作业运行在过高优先级、选择在一天中不恰当的时间运行大型作业），也包括由系统引起的问题（磁盘限额、CPU 记账、非必要的守护进程）。
- 对于将 Linux 用作 Web 服务器或者其他类型的网络应用服务器的情况，您可能想使用一种商业的负载平衡设备，比如 Cisco 的 Content Services Switch（www.cisco.com）、Foundry 的 ServerIron（www.foundry.com）或者 Nortel 的 Alteon Application Switch（www.nortelnetworks.com）来将流量分散到若干个系统上[1]。这些装置让多台物理服务器对外界呈现为一台逻辑上的服务器。它们根据可以由用户选择的算法（如“最快响应服务器”或“轮流法”）之一平衡负载。
- 这些负载平衡器还在一台服务器停机时提供了有用的冗余性。如果您的网点必须处理瞬间的大流量冲击，那么就确实有必要使用它们。
- 组织磁盘和文件系统以平衡负载，从而得到最大的 I/O 吞吐量。确保您为磁盘选择了合适的 Linux I/O 调度方法（参见 25.3.4 节了解详情）。对诸如数据库等特定的应用来说，可以利用多磁盘技术（如条带式的 RAID）优化数据传输。可咨询数据库销售商得到建议。
- 注意不同类型的应用和数据库对应于多磁盘间不同的数据分布是很重要的。RAID 具有多种形式，您需要花些力气来确定哪种形式最适合您的特定应用。
- 监视网络，确保流量不会饱和并且出错率低。利用 19.4 节介绍的 **netstat** 命令可以获得大量的网络信息。还可参考本书的第 19 章。
- 辨认系统基本上不足以满足要求的情况。

上述步骤是按照效果从高到低的大致顺序排列的。添加内存和在多台服务器间平衡流量可以在性能上得到巨大提升。其他因素的效果则从显著提高性能到毫无结果都有。

[1] Linux Virtual Server 这个免费软件也是一种选择，但它有点儿不太稳定，从 linuxvirtualserver.org 可以得到它。

www.TopSage.com

优化软件的数据结构和算法几乎肯定会得到显著的性能增益。但是除非您对本地软件有扎实的基础知识，否则达到这种水平的设计能力往往超出了您的控制范围。

25.2 影响性能的因素

用户能觉察到的性能是由系统资源分配和共享的效率来决定的。“资源”的准确定义相当模糊。它可以包括诸如 CPU 芯片上被高速缓存的上下文（context）以及内存控制器地址表中的表项这样的内容。不过，粗略来说，只有以下 4 种资源对系统性能有显著影响：

- CPU 时间；
- 内存；
- 硬盘 I/O；
- 网络 I/O。

所有进程都要消耗一部分系统资源。如果在激活的进程各自获得所需的资源后仍有剩余，则系统的性能大致上能有多好就有多好了。

如果没有足够的系统资源进行周转，那么进程必须轮流使用资源。此时不能立即访问其所需资源的进程，必须等着，什么也不干。进程所花的等待时间量是衡量系统性能降低的基本指标之一。

CPU 时间是最易测量的系统资源之一。CPU 的数据处理能力基本上是恒定不变的。在理论上应该是 CPU 周期的 100%，但由于开销和低效使得实际量更可能是 95%。使用 90%以上 CPU 周期的某个进程可认为是受限于 CPU（CPU-bound）的进程，它消耗着系统可用计算能力的大部分。

许多人认为 CPU 速度是影响系统整体性能的最重要因素。在系统其他资源均是无限量的，或者在特定类型的应用（例如数值模拟）的前提下，更快的 CPU 确实会带来明显的差异。但在日常现实中，CPU 的速度相对并不重要。

Linux 系统中最常见的性能瓶颈是磁盘的实际带宽。由于硬盘是机械系统，所以定位一个磁盘块、读取其中的内容以及唤醒等待它的进程这一系列操作需要花费若干毫秒的时间。这个数量级的延迟使得所有其他性能降低因素都黯然失色。每次磁盘访问都会导致相当于数百万条 CPU 指令的停顿。

由于 Linux 提供虚拟内存，因此磁盘的带宽和内存直接相关。在 RAM 数量有限的重负载系统中，内核不得不把一个内存页写入磁盘以获得一个新的虚拟内存页。遗憾的是，这意味着使用内存往往和使用磁盘的开销一样大。由臃肿的软件导致的调页（paging）在大多数工作站上是性能的头号敌人。

网络带宽由于包含延迟，所以在许多方面都与磁盘带宽类似。不过，因为网络涉及整个计算机群体而不仅仅是单台计算机，所以不太典型。网络也容易受到硬件问题和服务器过载的影响。

25.3 全面检查系统性能

大多数性能分析工具给出的是某个时间点上的情况。但是，负载的数值和特性整天都在不断变化。在采取措施之前应保证采集了代表性的数据。只有经过长期的数据采集（一个月或者更长时间），有关系统性能的最佳信息往往才能显现出来。在使用高峰期间采集数据尤其重要。往往只有在机器处于重负载的时候，才能看出资源的限制和系统的错误配置。

25.3.1 CPU 使用情况分析

您可能要采集 3 种 CPU 数据：总体利用率、平均负载和每个进程的 CPU 消耗量。总体利用率能

www.TopSage.com

有助于确定系统中CPU的运行速度是否是系统的性能瓶颈。平均负载可让您对系统的总体性能有一个印象。每个进程的CPU消耗量数据能够让您搞清占用系统资源过多的进程。

您可以利用 **vmstat** 命令获得汇总信息。**vmstat** 有两个参数：每行输出需监视系统的秒数，以及提供的报告数。如果您没有指定报告的行数，那么 **vmstat** 就会一直运行，直到您按下<Control-C>为止。**vmstat** 返回的第一行数据提供了自系统引导起来以后的平均值。随后的各行是在上一个采样期内的平均值，默认的采样时间为5秒。例如：

```
$ vmstat 5 5
procs  -----------memory----------  ---swap-------io----  --system--  ----cpu----
 r  b  swpd     free  buff     cache  si  so    bi  bo    in    cs  us  sy  id  wa
 1  0   820  2606356  428776  487092   0   0  4741  65  1063  4857  25   1  73   0
 1  0   820  2570324  428812  510196   0   0  4613  11  1054  4732  25   1  74   0
 1  0   820  2539028  428852  535636   0   0  5099  13  1057  5219  90   1   9   0
 1  0   820  2472340  428920  581588   0   0  4536  10  1056  4686  87   3  10   0
 3  0   820  2440276  428960  605728   0   0  4818  21  1060  4943  20   3  77   0
```

用户时间、系统时间和空闲时间分别在右边us列、sy列、id列和wa列中显示。用户时间数值较大一般表明计算机处于运算状态，而系统时间数值较大表明进程正在做大量系统调用或执行I/O操作。

一种我们已经使用多年、适用于大多数系统的粗略规则是系统中50%的非空闲时间将用于用户空间，而另外50%用于系统空间，同时总体的空闲百分比不应该为0。如果您把一台服务器专门用于一种CPU密集型应用，那么大部分时间应该都花在用户空间里。

cs列显示的是每个时间段中上下文切换的次数，也就是由内核切换当前运行进程的次数。每个时间间隔内的中断数（通常由硬件设备或者内核组件产生）在in列显示。cs或者in的数值极高一般表示硬件设备的配置或者运行有错误。其他列有助于分析内存和磁盘，我们随后讨论它们。

CPU的长期统计平均值能让您判断是否有完全足够的CPU处理能力来周转。如果CPU经常花一部分时间在空闲状态，则说明有空闲的CPU周期。此时更换速度更快的CPU虽然可以加快单个进程的执行速度，但对系统的总体吞吐量并无太多改善。

从这些例子可以看出，CPU主要是在大量使用和完全空闲这两种状态间来回变化的。因此，重要的是按一段时间内的平均值来观察这些数据。监视的时间间隔越小，所得结果的一致性就越差。

在有多个处理器的计算机上，大多数Linux的工具都会给出所有处理器上总的处理器平均统计数据。**mpstat** 命令能够给出针对单个处理器但是类似于 **vmstat** 的输出结果。**-P** 标志能让您指定一个要给出报告的特定处理器。**mpstat** 对于调试支持SMP（symmetric multiprocessing，对称多处理机技术）的软件来说很有用处——它也能显示出使用多处理器的系统效率是否高（或者低）。

对于只有一个用户的工作站来说，CPU通常有99%的运行时间处于空闲状态。因此当您在图形显示上滚动一个窗口时，CPU会有短暂的满载。在这种情况下，长期的CPU平均利用率也没有意义了。

第二种有助于刻画系统负载特征的CPU统计数据是“平均负载”，也就是可运行进程的平均数。它确实让您有了对CPU这个大饼被划分了多少块的清楚认识。用 **uptime** 命令可以获得负载的平均值：

```
$ uptime
11:10am up 34 days, 18:42, 5 users, load average: 0.95, 0.38, 0.31
```

给出的3个数值分别对应系统在5分钟、10分钟和15分钟内的平均负载值。一般说来，平均负载越高，系统的合并性能就越重要。如果系统中只有一个可运行的进程，那么该进程通常只受一种系统资源的限制（往往是磁盘带宽或CPU）。对一种系统资源的峰值需求成为性能的决定性因素。

当有多个进程共享一个系统时，负载可能分布得更均匀，也可能不是这样。如果系统中的所有进程均需要消耗CPU、磁盘和内存，那么系统性能就不大可能受一种资源限制的控制。在这种情况下，参考诸如总体CPU利用率这样的平均指标就变成最重要的了。

www.TopSage.com

一般来说，Linux 系统在平均负载达到 3 的时候就处于繁忙状态，而且不能很好地处理平均负载超过 8 的情况。这个量级的平均负载则提醒您，应该开始看看有没有能人为地分配负载的办法，比如要求用户在夜里运行处理时间较长的进程或者使用 **nice** 命令设置进程的优先级。有关优先级的详细内容请参考 4.1.5 节。

系统的平均负载值还是一个优秀指标，可以作为系统基准数据的一部分进行跟踪。如果知道您的系统在普通工作日中的系统平均负载，而在有问题的一天也处于同样的范围，这就是暗示您应该看看别的方面来确定性能问题（比如网络）的原因。如果平均负载超出预期的正常水平，则表明您应该看看 Linux 系统自身所运行的进程。

观察 CPU 使用情况的另一种方法是运行带 **aux** 参数的 **ps** 命令，它能让您看到每个进程正在使用多少 CPU。在比较繁忙的系统中，CPU 的至少 70%有可能只被一两个进程占用（记住 **ps** 命令本身也要消耗 CPU）。推迟执行这些极其耗费 CPU 的进程或者降低这些进程的优先级会让其他进程更容易用上 CPU。

ps 命令有一种优秀的替代程序叫做 **top**。**top** 所提供的信息和 ps 的大致相同，但是是以“动态”的格式输出的，这让您可以监视系统随时间变化的状态[2]。参考 4.8 节了解有关 **top** 的更多信息。

25.3.2 Linux 的内存管理

在我们讲述 Linux 内存管理的特别之处以前，值得再交待一下，Linux 的发展演变速度很快，远远超过了几乎其他任何操作系统。Linux 的内存管理系统已经处在一个特别迅速的变迁时期有好多年了，甚至连 Linux 的标准都是这样。

2.6 版内核里包含了一个得到极大改进的 VM 系统[3]。但是在我们在本章所讨论的层面上，内部版本到底是哪个并不真的有关系。不管您正在用哪个特定版本的内核，这些概念都适用。

和 UNIX 系统一样，Linux 也是按照称为“页”（page）的单元来管理内存的。目前在 PC 硬件上，页的大小为 4KB[4]。Linux 内核在进程需要内存的时候，分配给它们虚拟页。每个虚拟页都被映射到实际的存储器上，即 RAM 或者磁盘上的“备份存储”空间（备份存储通常是在交换区内的空间，但对于包含可执行程序代码的页来说，备份存储是原来的可执行文件）。Linux 使用一个“页表（page table）”来跟踪这些虚拟页同实际存储页之间的映射关系。

Linux 用交换空间（swap space）来增加实际 RAM 的大小，有效地向进程提供它们所需要的内存。进程都以为它们的虚拟页映射到了实际的内存上，所以 Linux 总是忙于在 RAM 和交换区之间来回换页。这种活动称为调页（paging）[5]。

Linux 试图对系统的内存进行管理，以使最近访问过的页保存在内存中，并将活动较少的页换出到磁盘。由于近期最少使用（least recently used）的页是轮换到磁盘上的页，所以这又称为 LRU 系统。在实际中，让内核跟踪所有的内存引用的话，效率极低，所以 Linux 采用一个页高速缓存（page cache）来决定换出哪些页。这种系统比真正的 LRU 系统开销低得多，但取得的结果类似。

Linux 跟踪着每个虚拟内存页的“年龄”。Linux 每次考察一页，发现它已经在最近被引用过的时候，它就累加这页的年龄（“年龄”这个词有点儿误导性，因为它的值表示的是访问是否频繁，最近是否访问过。年龄的值越大，这页就越新）。同时，Linux 运行 **kswapd** 进程，它定期把没有被引用过的

[2] 频繁刷新 top 的输出本身就要消耗大量的 CPU 资源，因此应谨慎使用该程序。

[3] 2.6 版内核有一个选项，就是不用虚拟内存来运行系统——这个选项对于嵌入式系统特别有用。如果您现在不带虚拟内存运行，那么就必须保证有足够的物理内存来满足所有可能的需要。

[4] 您也可以通过设定 2.6 版内核里的 HPAGE_SIZE 来使用 4MB 的内存页。

[5] 注意，“调页”和“交换”实现的方式相同，当页不在内存里时都使用同一个“交换”区保存页。

www.TopSage.com

页的年龄值降低。

内核维护着几个内存页的列表。年龄比0大的页被标记为“活动（active）”，并且保存在页高速缓存“活动列表（active list）”之中。如果某页的年龄达到0，**kwapd** 就把这页交换到“不活动列表（inactive list）”中。**kswapd** 从页表中删除不活动的页，认为它们符合条件，应该被调到磁盘上去。虽然不再能立即通过页表访问到不活动的页，但是如果需要，内核会从内存或者磁盘上恢复它们，把它们送回到页表中。

当内存不足时，内核就试着猜测不活动列表中的哪些页是近期最少使用（LRU）的。如果那些页已经被某个进程修改过了，那么 Linux 就认为它们是“脏的（dirty）”，在可以重新使用这部分内存之前，必须把它们换出到磁盘上。已经经过这番处理的页，或者一直没有变“脏”的页则是“干净的（clean）”，可以由 Linux 重新用到别处去。

在某个进程引用了处于不活动列表里的某一页时，内核就把它的内存映射关系返回到页表中，重设这页的年龄，把它从不活动列表转入活动列表。已经被写入磁盘的页必须在重新激活它们之前被换入内存。当一个进程引用了一个内存中的不活动页时，就发生了一次“软的页缺失错（soft fault）”，而“硬的页缺失错（hard fault）”是因为引用了一个不在内存（被换出了）中的页。换句话说，硬缺失错要求从磁盘读回一页，而软缺失错不需要。

内存的需求量是在不断变化的，所以内核能以不同的速度运行 **kswapd** 的页老化算法。当内存的需求量极大时，算法运行得更为频繁，页也必须要更常引用才能避免因为它们的年龄降至0而被换出。

为了从不活动的页中找出活动的页，虚拟内存（virtual memory，VM）系统要靠页被放入不活动列表的时间到它实际被换出（以便把活动页与不活动页分开）的时间之间的滞后来判断。因此，VM 系统必须对将来的调页活动进行预测，以决定运行 **kswapd** 的频度。如果 **kswapd** 运行的频度不够，那么干净的不活动页就没有能满足需要的那么多。如果 **kswapd** 运行得太过频繁，那么内核就花费了过多的时间处理软的页缺失错。

即使调页算法有预测性，可是也没有必要在换出事件和正在运行的进程请求分配页之间有一一对应的关系。系统的目的在于，保持足够多的空闲内存待命，在实际中，进程不必每次在分配新的内存页时都要等着其他页被换出。如果系统繁忙的时候调页次数激增，那么有更多的 RAM 或许效果更好。

您可以调节内核的“交换度”参数（**/proc/sys/vm/swappiness**），给内核一个提示，告诉它在发生内存不足的时候，应该以多快的速度从一个进程那里收回物理内存页。在默认情况下，这个参数的值为60。如果您把它设为40，那么内核只有在它已经用完了所有可能的内存之后，才会采取把已经分配给一个进程的内存页收回的办法。如果这个值设为60以上（最大值为100），那么内核在缺少内存的时候从一个进程那里收回内存页的可能性就更大。无论如何，如果您发现自己禁不住要去修改这个参数，或许就是该给系统增购内存的时候了。

如果内核用完了 RAM 和交换区，那么所有的 VM 也就耗尽了。Linux 使用“内存不足杀手（out of memory killer）”来处理这种情况。这一功能会选择和杀死一个进程，从而释放出内存。虽然内核企图杀死的是系统上最不重要的进程，但内存不足的情况还是一定要避免。在这种情形下，很有可能系统资源的很大一部分都用在了内存管理上，而不是用在有用的工作上。

即使进程以较低的 CPU 优先级运行，也可暗中窃取内存页。例如，如果在工作站中以极低的优先级（高 nice 值）运行 SETI[6]，同时还从某个终端窗口阅读电子邮件。当您暂停下来阅读一则消息时，整个系统的 CPU 使用降到零，从而允许模拟程序运行。它将自己的所有页换入内存，却迫使您的 shell、窗口服务器、邮件读取器以及终端模拟器被换出去。当您阅读下一则消息时，便会出现延迟，因为一大块系统内存要更换内容。在现实生活中，高的 nice 值也不能保证进程不会引起性能问题。

[6] 它代表 Search for Extraterrestrial Intelligence（寻找外星人）。

www.TopSage.com

25.3.3 内存使用情况分析

内存活动基本上可以用 3 个数字来量化：活动虚拟内存的总量、交换(swapping)率和调页(paging)率。其中第一个数字表明内存的总需求量，后两个数字表明那些内存中有多少比例正处于使用之中。目标是要减少内存活动或增加内存量，直到调页率保持在一个可接受的水平上为止。偶尔的调页是不可避免的，因此不要为了要完全消除调页而操心。

您可以使用 **free** 命令来判断当前投入使用的内存和交换空间的数量。带**-t** 标志执行这条命令会自动计算出虚拟内存的总量。

```
$ free -t
             total       used       free     shared    buffers     cached
Mem:        127884      96888      30996      46840      57860      10352
-/+ buffers/cache:      28676      99208
Swap:       265032       3576     261456
Total:      392916     100464     292452
```

free 列表明了系统上空闲列表以 KB 计的大小，这个值低于系统总内存量的 3%一般表明有问题。您可以使用 **swapon** 命令来准确地判断出正在用哪些文件和分区作为交换空间。

```
$ swapon -s
Filename                        Type            Size    Used    Priority
/dev/sdb7                       partition       265032  3576    -1
```

这个系统将磁盘分区 sdb7 用作交换空间。既然调页活动可以占用大量的磁盘带宽，所以如果可能的话，把您的交换空间分布在多个物理磁盘上几乎总是比较好的做法。使用多个交换磁盘比起一个磁盘来说，在处理交换数据上所花的时间降低了，而且还增加了交换存储的带宽。

在传统的 UNIX 系统上，**vmstat** 命令提供了有关调页和交换的信息。不过，大多数 Linux 发行版本都带的 **procinfo** 命令也是一种良好的信息来源。虽然我们举例的所有系统都带有 **procinfo** 命令，但它却不一定包含在默认安装里，所以您可能需要手工加上 **procinfo** 软件包。

procinfo 并没有什么特别的途径来得到有关系统的信息，它只是把从文件系统**/proc** 中的文件里得到的数据以比较好的格式显示出来而已。没有它的话，要解读**/proc** 下的文件内容真是一项严峻的挑战。运行 **procinfo -n5** 能以 5 秒为间隔连续刷新输出结果。要了解有关**/proc** 文件系统的更多信息，请参考 28.4 节。

```
$ procinfo
Linux 2.6.9-34.EL (root@bull) (gcc 3.4.3 20050227 ) #1 1CPU [main]

Memory:      Total        Used        Free      Shared     Buffers
Mem:        463728      371512       92216           0      144042
Swap:      1866470         218     1626252

Bootup: Sat Jul 29 08:47:20 2006   Load average: 0.08 0.06 0.00

user   :          1:20:32.12   1.3%     page in :    0
nice   :          0:02:36.30   0.0%     page out:    0
system:           3:06:46.90   0.8%     swap in :    0
idle   :       1d 7:23:49.50  97.9%     swap out:    0
steal  :          0:00:00.00   0.0%
uptime:        4d 1:51:43.64              context :    55465717

irq  0:   5748875398  timer          irq  8:          3  rtc
irq  1:            3                 irq  11:  75300822  eth0
irq  2:            0  cascade [4]    irq  12:         3
irq  4:            4                 irq  14:    7529735  ide0
```

www.TopSage.com

procinfo 输出的信息有一些和 **free**、**uptime** 和 **vmstat** 输出的信息重复了。此外，**procinfo** 提供了有关内核版本、内存调页、磁盘访问以及 IRQ 分配的信息。可以使用 **procinfo -a** 看到/**proc** 文件系统里的更多信息，其中包括内核的引导参数、内核的可加载模块、字符设备和文件系统。

在和内存有关的列中，任何明显的不一致性都会很容易让人产生错觉。有些列以页为计算单位，而其他的列则以 KB（千字节）为计算单位。所有的值都是四舍五入的平均值。而且，有些是标量的平均值，其他则是增量的平均值。例如，您无法从当前的 Free 和调页信息计算出下一个 Free 的值，因为决定下一个 Free 的平均值的调页事件还没有发生呢。

使用 `page in/out` 和 `swap in/out` 字段来评估系统的调页和交换行为。一次调入（page-in）并不一定表示从交换区域恢复的一页。它可以是从一个文件系统正被调入的可执行代码，或者一个正被复制的写时复制（copy-on-write）页，两者都是正常的事件，不一定表示内存短缺。另一方面，调出（page-out）一定表示数据被内核强行弹出内存。

如果系统持续发生调出（page-out）事件，那么就需要更多的内存。但是如果只是碰巧才发生调页，而且没有造成烦人的系统暂停现象或者招致用户抱怨，您就可以不去管它。如果系统发生调出（page-out）的情况处在中间的某个程度上，那么下一步的分析要取决于您是要为交互性能做优化（例如，一台工作站），还是要配置一台有许多并发用户的机器（例如，一台服务器）。

如果有一半的操作是调出（page-out），您可以估算出每 50 次调出操作造成大约 1 秒的延迟。如果为了让您滚动一个窗口必须出现 75 次调出，那么就要等候大约 1.5 秒。研究界面的人员采用这样一种原则，当响应时间超过 0.7 秒时，普通用户就会感觉到系统“慢”了。

值得指出的还有一点，**procinfo** 提供的一些 CPU 信息是用 **vmstat** 或者 **uptime** 看不到的。除了报告 5、10 和 15 分钟的平均负载之外，**procinfo** 还列出了瞬时正在运行的进程数量、进程总数以及最近运行的进程的 PID。例如，下面节录了从一台繁忙的服务器上运行 **procinfo** 的结果：

```
Load average: 2.37 0.71 0.29 3/67 26941
```

这台服务器总共有 67 个进程，其中有 3 个可以运行。最近运行的进程是 PID 26941（在本例中，它是 shell 进程，**procinfo** 从这个 shell 里执行）。

25.3.4 磁盘 I/O 分析

您可以使用 **iostat** 命令监视磁盘的性能。和 **vmstat** 一样，它接受的可选参数包括以秒为单位的时间间隔以及重复次数，而其输出的第一行是系统自引导以来的汇总。和 **vmstat** 命令一样，**iostat** 命令也能告诉您 CPU 的时间是怎样用掉的。下面是一个例子，其中省略了与磁盘无关的一些输出。

```
$ iostat
...
Device:    tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdisk0    0.54         0.59         2.39     304483    1228123
hdisk1    0.34         0.27         0.42     140912     216218
hdisk2    0.01         0.02         0.05       5794      15320
hdisk3    0.00         0.00         0.00          0          0
```

iostat 从文件系统/**proc** 收集信息，然后为系统里的每个物理磁盘产生一行输出。遗憾的是，Linux 只保留最少的磁盘统计信息，而且即便是保留下来的信息作用也很有限。每个磁盘都有 `tps`、`Blk_read/s`、`Blk_wrtn/s`、`Blk_read` 和 `Blk_wrtn` 列，分别表示每秒的 I/O 传输次数、每秒读取的块数、每秒写入的块数、读取的总块数和写入的总块数。如果系统上的磁盘数量没有 4 个之多，那么会报告几个全零行。

www.TopSage.com

磁盘块一般的大小是 1K，所以很容易判断出以 KB（千字节）计算的实际磁盘吞吐率。另一方面，传输的定义则相当含糊。一次传输请求可以包括对几个扇区的若干次 I/O 请求，所以这个数据大多用于辨别发展趋势或者非常规的行为。

寻道的开销是影响磁盘驱动器性能的最重要因素。粗略估算一下，磁盘的旋转速度和与其相连的总线的速度的影响都相当小。现代的磁盘如果读取连续的扇区，每秒就能传送数 10MB 的数据，但是它们每秒只能执行 100～300 次寻道。如果每次寻道只能传送一个扇区，您就能很容易地发现该驱动器的吞吐量不足其峰值的 5%。

当寻道让磁头移动的距离较大时，其所耗费的时间也较长。如果磁盘具有多个文件系统分区，而且从每个分区随机读取文件，则不同分区之间的切换将使磁头来回移动较长的距离。另一方面，一个分区内的文件则彼此相对靠近。在给一个新磁盘划分分区时，您可能需要考虑这些隐含的性能因素，而将需要同时访问的文件归入同一个文件系统中。

为了真正获得最高的磁盘性能，应该把一起使用的文件系统放置在不同的磁盘上。尽管这依赖于总线结构和设备的驱动程序，但大多数计算机能够独立管理多个磁盘，这大大地增加了磁盘的吞吐量。例如，经常还是值得花时间将频繁访问的 Web 服务器数据和日志分置于多个磁盘中。

由于调页会使整个系统变慢，所以如果可能，将交换空间散布在多个磁盘上则特别重要。使用 **swapon** 命令就能支持这一配置。Linux 内核既能使用专用的交换分区也能使用格式化的文件系统中的某个交换文件。

现代的 Linux 系统能让您设置多个“基于内存的文件系统”，它们实质上就相当于 PC RAM 磁盘。它是一种特殊的驱动程序，做出来一个磁盘的样子，但实际上却把数据保存在高速的内存中。许多地方将 RAM 磁盘用于它们的**/tmp** 文件系统或者其他繁忙的文件，比如 Web 服务器的日志文件或者电子邮件池。使用 RAM 磁盘会减少供普通使用的内存数量，但是它在表面上加快了临时文件的读写速度。一般而言，它是一种划算的做法。

大多数 Linux 内核都已经把对 RAM 磁盘的支持编译进来了。在支持 RAM 磁盘的系统上，**/dev** 目录包含多个 RAM 磁盘设备文件，比如**/dev/ram0** 和**/dev/ram1**。设备的数量可能有不同，但是通常至少有 5 个。

要使用一个 RAM 磁盘，首先要在一个没有用过的 RAM 磁盘设备上格式化文件系统，然后把它安装到一个现有的目录上。

```
# mke2fs /dev/ram12
# mount /dev/ram12 /tmp/fastdisk
```

默认的 RAM 磁盘大小仅为 4MB，这对于**/tmp** 来说实际上不够大。遗憾的是，改变这个大小的过程有点儿痛苦，因为它被设为一个内核变量。您要么在引导加载程序的配置里加上像 `ramdisk_size=100000` 这样的一行，在引导时刻把新的值传入内核（大小是以 1K 的块数来算），要么就把 RAM 磁盘的驱动程序设置为一个动态可加载的模块。在后一种情况下，您要向**/etc/modprobe.conf** 文件里加进必要的参数，或者把它作为 **insmod** 的参数传递给内核。

有些软件包由于拖延了基本操作从而降低了系统的性能。磁盘限额和 CPU 记账就是这方面的两个例子。磁盘限额要求在写文件及删除文件时更新磁盘使用情况的汇总。CPU 记账功能则在每个进程结束时把数据记录写入记账文件中。磁盘的高速缓存有助于减轻这些功能的影响，但它们可能仍然对性能有显著影响，除非您确实需要使用它们，否则不要启用它们。

25.3.5 选择磁盘 I/O 调度器

I/O 调度算法在各个进程竞争磁盘 I/O 的时候担当了裁判的角色。它要对请求的次序和时机做最

www.TopSage.com

优化处理，以求取得尽可能最好的整体 I/O 性能。

在 2.6 版内核里有 4 种不同的调度算法。您可以自行选择。遗憾的是，调度算法要由内核参数 **elevator=*algorithm*** 来设定，所以只能在系统引导的时候设置。通常在 **grub.conf** 这个文件里完成设置。

下面列出这 4 种调度算法。

- CFQ（Completely Fair Queuing，完全公平的排队）（**elevator=cfq**）：这是默认算法，对于通用服务器来说通常是最好的选择。它试图均匀地分布对 I/O 带宽的访问。
- Deadline（**elevator=deadline**）：这个算法试图把每次请求的延迟降至最低。该算法重排了请求的顺序来提高性能。
- NOOP（**elevator=noop**）：这个算法实现了一个简单 FIFO 队列。它假定 I/O 请求由驱动程序或者设备做了优化或者重排了顺序（就像一个智能控制器完成的工作那样）。在有些 SAN 环境下，这种选择可能是最好选择。
- Anticipatory（**elevator=as**）：这个算法推迟 I/O 请求，希望能对它们进行排序，获得最高的效率。对于桌面工作站来说，这个算法可能是一个不错的选择，但对服务器则很少会理想。

判断出哪一种调度算法最适合于您的环境——您或许会 4 种都试试——您就能直接对磁盘 I/O 的性能施加影响了。

25.3.6 sar：连续采集和报告统计信息

在 Linux 上已经有一种常见的性能监视工具，它就是 **sar**。这个命令源自于早期的 AT&T UNIX，但它已经为在 Linux 下使用而经过了彻底地改写。

sar 乍看起来似乎与 **proinfo**、**vmstat** 和 **iostat** 显示的信息雷同。不过，它们之间有一个重要的区别：**sar** 既能报告当前数据，也能报告历史数据。

sar 不带选项执行会以 10 分钟为间隔报告自午夜起当天的 CPU 利用率，如下面所示。脚本 **sal** 收集了这些历史数据，它是 **sar**（有时称为 **sysstat**）软件包的一部分，必须设置为定期从 **cron** 里运行。**sar** 把自己收集的数据以二进制格式保存在**/var/log/sa** 里。

```
$ sar
Linux 2.6.9-11.ELsmp (bull.atrust.com)  08/04/2006

12:00:01 AM       CPU     %user     %nice   %system   %iowait     %idle
12:10:01 AM       all      0.10      0.00      0.04      0.06     99.81
12:20:01 AM       all      0.04      0.00      0.03      0.05     99.88
12:30:01 AM       all      0.04      0.00      0.03      0.04     99.89
12:40:01 AM       all      0.09      0.00      0.03      0.05     99.83
12:50:01 AM       all      0.04      0.00      0.03      0.04     99.88
01:00:01 AM       all      0.05      0.00      0.03      0.04     99.88
```

除了 CPU 的信息之外，**sar** 还能报告指示磁盘和网络活动的参数。使用 **sar-d** 能够得到当天磁盘活动的情况汇总，而 **sar -n DEV** 则能给出网络接口的统计信息。**sar-A** 可以报告所有的信息。

sar 也有一些局限性，但是它对于快速粗略了解历史信息来说很适用。如果您在慎重选择长期监测性能的工具，那么我们建议您配置诸如 Cacti 这样的数据采集和图示化平台。Cacti 用于网络管理，但它实际上能绘制任何的系统数据图，如 CPU 和内存信息。参考 19.10.3 节，看看 Cacti 画出的图的例子。

25.3.7 oprofile：全面的剖析器

oprofile 是一种功能异常强大的集成系统剖析器，它适用于运行 2.6 版以上内核的 Linux 系统。它

www.TopSage.com

能剖析 Linux 系统的所有组成部分：软硬件中断处理程序、内核模块、内核本身、共享库和应用软件。

如果您有很多富余的时间，而且想准确掌握系统资源正在使用的情况（小到最细枝末节的地方），那么就可以考虑运行 **oprofile**。如果您正在开发自己的内部应用软件或者内核代码，这个工具尤其有用。

在 **oprofile** 软件发布中有一个内核模块和一组用户级的工具，从 oprofile.sourceforge.net 可以下载到这个软件。



在 Red Hat 系统上，您还需要安装 **kernel-debuginfo** 这个软件包获得运行 **oprofile** 所需的文件。

25.4 求助！系统为何越来越慢

在前面几小节里，我们主要讨论了与系统平均性能相关的一些问题。对这些问题的长远解决办法一般是要采取调整或升级配置的方式。

不过，您会发现，即使是配置合理的系统有时也会出现比正常情况运行速度慢的问题。幸好这些暂时性的问题往往十分易于诊断。90%的情况是由于某个贪婪的进程消耗了过多的 CPU 资源或者占用了太多的磁盘或网络带宽，致使其他进程暂停运行。偶尔也是恶意进程占用可用资源，故意减缓系统或者网络的运行，这种做法叫做“拒绝服务（denial of service）”或者 DoS 攻击。

您甚至不用运行诊断命令，也经常能辨别出哪种系统资源被大量占用。如果感觉系统运行“粘滞”或者您听到磁盘疯狂运转，则问题最有可能是磁盘带宽或者内存不足[7]。如果感觉系统运行“缓慢”（每次都要花很长一段时间，应用程序不能“激活”），则问题可能出在 CPU 负载上。

诊断的第一步是运行 **ps auxww** 或者 **top** 以查找明显的失控进程。任何占用 50%以上 CPU 资源的进程都可能有故障。如果所有进程均未过多消耗 CPU 资源，则要检查有多少个进程至少占用了 10%的 CPU 资源。如果存在两个或三个以上这样的进程（不要计算 **ps** 本身），则表明系统的平均负载可能太高。这是造成系统性能降低的一个原因。用 **uptime** 可检查系统的平均负载，用 **vmstat** 或 **top** 可检查 CPU 是否处于空闲状态。

如果没有 CPU 争用的迹象，可以用 **vmstat** 或者 **procinfo** 查看目前处理的调页数量。所有的磁盘活动都是令人关注的：太多的调出（page-out）可能预示着内存争用，没有调页，但却有磁盘通信量可能意味着某一个进程正以持续读文件或写文件的方式独占了磁盘。

还没有一种直接的方法可以将磁盘操作和进程关联起来，但是 **ps** 能够缩小可能的怀疑对象的范围。任何产生磁盘通信量的进程必须使用一定数量的 CPU 时间。通常可以根据经验推断出活动进程中哪一个真正存在问题[8]。用 **kill -STOP** 可以挂起这个进程，验证上述推测的正确性。

假设您确实发现了某个特殊进程存在问题——那么您应该怎么做呢？通常情况下，什么也不做。有些操作只是需要大量的资源，势必会减缓系统的运行。这并不一定意味着它们是不合理的。通常可以接受的方法是用 **renice** 改变过多占用 CPU 资源的突出进程的 **nice** 值。但是要确保让进程的主人以后使用 **nice** 命令再改回来。有时候，对应用进行调优会大大降低程序对 CPU 资源的需求，采用定制的网络服务器软件（比如 Web 应用）之后，就特别容易看出这个效果来。

处理过多占用磁盘或者内存资源的进程就不能如此简单了。**renice** 命令一般说来没有帮助。您确

[7] 也就是说，应用程序间的切换将耗费很长的时间，但是在应用程序仅重复一项简单任务时系统的性能尚可接受。

[8] 较大的虚拟地址空间或驻留集过去是一种引起怀疑的征兆，但共享库已经使得这些数字的用处小了许多。**ps** 并不善于把系统范围内的共享库的开销和单个进程的地址空间分开。许多进程错误地表现出占用了数兆字节的活动内存。

www.TopSage.com

实可以把杀死或停止进程作为一种选择，但如果没有达到紧急情况，我们并不建议这样做。和对待过多占用 CPU 的情况一样，您可以采用技术性不那么强的解决方法，请该进程的主人以后再运行它。

Linux 内核允许通过 **setrlimit** 系统调用对一个进程消耗的物理内存加以限制。在 C shell 中采用其内置的 **limit** 命令即可获得这一功能。例如，命令：

```
% limit memoryuse 32m
```

将使用户随后运行的所有命令对物理内存的使用都限制在 32MB 以内。这个功能和用 **renice** 调整占用内存过多的进程大致等效。您可以巧妙地建议老是犯错的用户把这一行放入他们的**.cshrc** 文件中。

如果失控进程并不是导致系统性能降低的原因，则应对系统另外的两个可能因素进行考查。第一是网络过载。许多程序都与网络有着密不可分的联系，以至于很难分清系统性能的终点和网络性能的起点。关于网络监视工具的详细内容请参考本书第 19 章。

有些网络过载问题由于来去匆匆，因此很难进行诊断。例如，如果网络中的每台计算机都在每天的特定时间利用 **cron** 运行与网络相关的程序，就会经常出现一些短暂的但是明显的小故障。网络中的每台机器都将停顿约 5 秒的时间，然后上述问题又和出现时一样迅速地消失了。

与服务器相关的延迟是另一个引起系统性能危机的可能原因。Linux 系统将不断地向远程服务器查询 NFS、NIS、DNS 以及十几种其他设施中的任何一种。如果服务器死机或者其他方面的问题使得与它通信的开销很大，那么将会很快通过客户机系统反映出来。

例如，在一个繁忙的系统中，有些进程可能大约每隔几秒使用一次 **gethostent** 库例程。如果 DNS 的小故障致使该例程要花两秒才能完成，您就可能会感觉到在总体性能上的差异。DNS 转发和反向查询上的配置问题要为数量惊人的服务器性能问题负责。

25.5 推荐读物

EZOLT, PHILLIP G. *Optimizing Linux Performance*. Upper Saddle River, NJ: Prentice Hall PTR, 2005.

JOHNSON, S., ET AL. *Performance Tuning for Linux Servers*. Indianapolis, IN: IBM Press, 2005.

LOUKIDES, MIKE, AND GIAN-PAOLO D. MUSUMECI. *System Performance Tuning, (2nd Edition)*. Sebastopol: O'Reilly & Associates. 2002.

25.6 习题

E25.1 对下面每种情形的问题做一个合理的猜测。

a）在应用间进行切换时，磁盘狂转，但是没有明显的迟滞。

b）一个数值模拟程序占用的时间比正常值大，但是系统内存的大多数都是空闲的。

c）在一个繁忙的 LAN 上的用户抱怨说 NFS 访问缓慢，但是服务器上的平均负载却非常低。

d）运行一条命令（任何命令）经常提示说“内存不足”。

E25.2 在能用的 Linux 机器上运行 **procinfo** 命令，讨论输出结果。如果有什么资源显得使用量很大的话，它是什么资源？什么资源似乎没有用到？在您的分析中要包括 IRQ。

★E25.3 负载均衡能大大影响到从外界来看的服务器性能。讨论几种可以用于负载均衡的机制。

www.TopSage.com

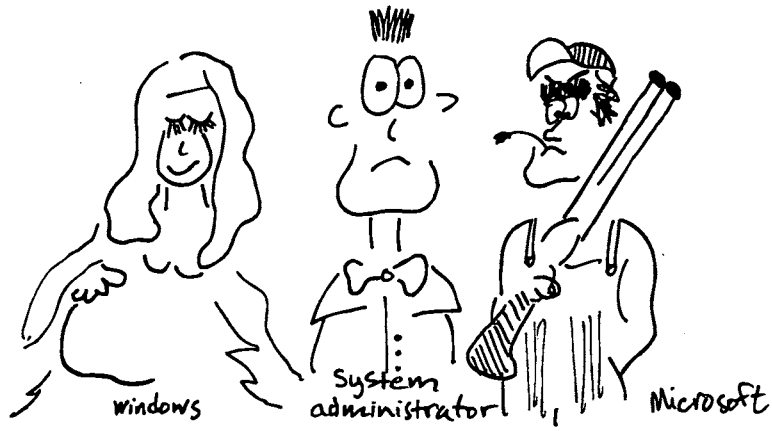
★E25.4 列出能够影响到性能的4种主要的资源。针对每种资源，给出一个应用的例子，这个应用可以轻而易举地致使那一资源被耗尽。讨论可以采用哪些方式来减轻与每种情况相关的压力。

★E25.5 通过 Web 和手册页，研究 **hdparm** 命令。它提供了哪些选项，可以测试访问磁盘的速度？**hdparm** 在某些情况下是怎样改善访问磁盘的速度的？

★★E25.6 选择两个使用了大量系统资源的程序。用 **vmstat** 和本章里介绍的其他工具剖析这两个应用程序。说明每个程序是因为做了什么才成为资源消耗大户的。用数据来支持您的结论。

www.TopSage.com

第 26 章　同 Windows 协作



您的环境非常有可能既包括微软的 Windows 系统，又包括 Linux 系统。如果是这样的话，这两种操作系统能以许多种方式彼此提供帮助。特别地，Windows 应用可以从 Linux 台式机运行，或者访问 Linux 服务器的打印机和文件。Linux 应用也能显示 Windows 台式机上的用户界面。

这两种操作平台各自都具有十分强大的功能并且可以一起合作。Windows 是一种流行而且功能全面的桌面平台，它能够架起跨越用户与墙上伸出的网线之间沟壑的桥梁。另一方面，Linux 则是一种稳固的、可扩展的基础设施平台。让我们和平共处，好吗？

26.1　从 Windows 登录到 Linux 系统

用户可能经常会发现他们自己既想好好用用 C shell 或 **bash**，又不想放弃桌上的 Windows 系统。从用户的角度来看，要达到上述目的，最简单的途径是使用微软随 Windows 带的 **telnet** 程序。尽管现如今大多数 Linux 系统出于安全原因，都已经（明智地）把系统里的 **telnet** 服务程序给关掉了。想要登录到 Linux 系统中的用户需要一种能够支持 SSH 的终端模拟程序。

Windows 上有几种 SSH 实现。我们最喜欢的是开放源代码的 PuTTY，它简单而高效。PuTTY 支持许多像 xterm 这样的纯终端应用程序具有的功能。对于 PuTTY，我们听到的唯一不满是它不能正确地处理多字节字符。给 Tera Term Pro 加上 TTSSH 插件可能更适合于亚洲语言的用户，虽然这样的配置只支持 SSH 协议第 1 版。您可以从下面的地址获得 PuTTY：

www.chiark.greenend.org.uk/~sgtatham/putty

www.TopSage.com

大学网 TopSage.com

和 Tera Term Pro：

hp.vector.co.jp/authors/VA002416/teraterm.html

SSH 也支持文件传输，PuTTY 包括两个针对此目的的命令行客户程序：**psftp** 和 **pscp**。恪守“绝不碰命令行”信条的 Windows 用户更喜欢 winscp.sf.net 的图形界面 WinSCP 客户端程序。

另一种可行的选择是安装更通用的 Cygwin 软件包（在 Windows 上的 UNIX），从 **rxvt** 中运行它的 SSH 工具程序。有关 Cygwin 的更多信息可参见 26.4 节的内容。

从 AppGate 可以获得 SSH 的 Java 实现，它叫做 MindTerm。个人使用该软件免费。它能运行在任何支持 Java 的系统上，并且能配置成多种不同的方式。

SSH 有个很有意思的功能，就是它能在客户机和服务器之间来回定向（forwarding）TCP 端口。例如，这个特性能让您在客户机上建一个本地端口，把传入的连接定向到一台只能从服务器访问到的机器上的一个不同的端口。虽然这个特性带来了很多新的可能性，但是也有潜在的危险，您在让服务器能通过 SSH 访问的时候必须注意这一点。还好，在服务器端可以禁用端口定向（port forwarding）功能，限制 SSH 只用作终端访问和文件传输。

26.2 远程桌面访问

Linux 上的图形桌面是免费的 X 窗口系统（X Window System），X 与微软的 Windows 没有任何关系。X Windows 由麻省理工学院（MIT）在 20 世纪 80 年代中期开发，并且已被所有的 UNIX 工作站制造商和 Linux 发行版本选作他们的一种标准，但是 X 的稳定基础版本最后是第 11 版，这个版本在 20 世纪 90 年代初期就发布了。该协议的版本号加在 X 后面构成 X11，正是它最常见的名字（在本章里和现实中一样，“Windows”这个称呼本身始终都指微软的 Windows 系统）。

X11 是一种客户机/服务器系统。X 服务器负责在用户的屏幕上显示数据，并获得用户鼠标和键盘的输入。它通过网络与客户机应用程序进行通信。服务器和客户机不必在同一台机器上运行。

本书第 22 章更详细地讨论了 X Windows 的系统结构。

26.2.1 在 Windows 计算机上运行 X 服务器程序

X11 是一种功能很丰富的协议，这些年来融入了许多扩展。因此，X 服务器的实现相当复杂。尽管如此，如今在几乎每一种操作系统上，都有 X 服务器的实现。X Windows 本身与操作系统无关，所以，运行在 Linux 上的 X11 客户机能够在微软 Windows 上运行的 X 服务器上显示出来，并且让用户控制它们，就好像用户正坐在系统控制台上一样。

遗憾的是，X 协议的最初设计者们没有多考虑安全性。连接到 X 服务器上的每个程序都能读入键盘上输入的任何内容，看到屏幕上显示的任何东西。让情况更加糟糕的是，远端的程序在访问 X 服务器的时候甚至不必显示一个窗口，它们可以悄悄地藏在后台。

随着时间推移，人们提出了几种加强 X11 安全性的方法，但是这些方法都显得有些复杂。保底的做法是禁止所有远程的到 X 服务器的连接，能绝对掌握情况的连接除外。大多数 X 服务器都配置为默认拒绝远程连接，所以只要不运行 **xhosts** 程序（或者其等价程序）允许远程访问，就应该是安全的。

遗憾的是，当您想要在 Linux 上运行程序，而在 Windows 上显示其界面的时候，允许 X 的远程访问就正好是您需要做的。这样一来，该如何运行一个远程应用，又不会允许远程访问 X 服务器呢？最常见的办法就是使用 SSH 协议专为支持 X11 而设计的特性。这个方案在运行于远程主机上的 X 客户机和本地 X 服务器之间创建一个安全隧道。在远程主机上启动的程序将自动在本地机器上显示出来，但由于 SSH 的神奇功能，本地 X 服务器会以为它们就是从本地启动的。参见 20.11.3 节了解有关 SSH

www.TopSage.com

的更多知识。

注意，X forwarding 只有在 SSH 服务器和 SSH 客户机两边都启用了 X forwarding 功能时才运行。如果您在 Windows 上使用 PuTTY 的 SSH 客户机，只要在其设置界面里启动 X forwarding 功能即可。在 SSH 服务器端（也就是说，X11 的客户机端，Linux 主机），要确保在文件**/etc/ssh/sshd_config** 里有这么一行：

```
X11Forwarding yes
```

如果您修改了 SSH 服务器的配置，要重新启动 **sshd** 进程激活这个新配置。

虽然 Apple 提供了一个用于 Mac OS X 的免费 X 服务器，但是很遗憾，微软没有提供这样的软件。不过，Cygwin 项目里有一个免费的 X 服务器，配置好之后效果非常好。遗憾的是，对于不知道如何在 Linux 上配置 X 服务器的人来说，它的配置相当难弄。Windows 上商业的 X 服务器有 eXceed 和 X-Win32。这两种软件价格不菲，但配置却简单得多。

26.2.2 VNC：虚拟网络计算技术

在 20 世纪 90 年代末，英国剑桥 AT&T 实验室里的几个人开发了一个用于远程桌面访问的系统，叫做 VNC。他们的思想是把“哑”终端的简单性同现代的窗口系统相结合。与 X11 相比，VNC 协议不处理单个应用。相反，它创建一个完整的虚拟桌面（或者提供对现有桌面的远程访问）。在 VNC 下，在中央主机上运行一个特殊的 X11 服务器，并且用一个观察器应用来访问那个服务器。

AT&T 公布 VNC 软件的时候采用了一种慷慨的开放源代码许可证。这就能让其他人参与进来，开发出更多的服务器和观察器的实现，并且改进协议，降低消耗的带宽。现如今，在大多数能以某种方式提供图形显示功能的设备上都有相应的 VNC 观察器。UNIX/Linux 和 Windows 的 VNC 服务器也很普及。

Linux 的 VNC 服务器实现实质上是一个图形适配模拟器，它以插件形式插入到 X.Org 的 X 窗口系统里。从用户的 Linux 账号运行 **vncserver**，就能创建一个新的虚拟桌面，这个桌面就运行在 Linux 机器上。于是您可以使用一个 VNC 观察器远程访问那个桌面。

VNC 协议是无状态协议，而且基于位图。因此，观察器能自行连接和断开。几个观察器甚至可以同时访问同一个 VNC 服务器。对于远程支持和培训配置来说，最后这种特性格外有用。它还能在做系统管理工作的时候共享对控制台的访问。

Windows 上的 VNC 服务器一般不会再创建桌面，它们只是把标准的 Windows 桌面按屏幕上显示的样子来输出。这项技术的主要用途是远程支持。

现如今，VNC 的初创人员开办了他们自己的公司 RealVNC。Ultr@VNC 项目集中在 Windows 领域里开发一种速度很快、功能丰富且基于 Windows 的 VNC 服务器实现，而 TightVNC 则致力于改进压缩率。这几个开发小组会彼此沟通，所以在各种实现上的功能都交织了起来。

VNC 协议在设计上就考虑到了扩展性。观察器和服务器的所有组合都能行，它们会选择彼此都能理解的最好协议变体。特定于实现的功能（比如文件传输）通常只能在运行的服务器和客户机都来自于相同的项目时才能用。

26.2.3 Windows RDP：远程桌面协议

从 Windows 2000 Server 开始，每台 Windows 机器在技术上都能让多个用户同时从远程以图形方式访问系统。实现远程访问的系统组件称为远程桌面（Remote Desktop），它使用 RDP（Remote Desktop Protocol，远程桌面协议）在客户机和服务器之间进行通信。

www.TopSage.com

有了合适的 PC 硬件，您可以轻而易举地在一台 Windows 2003 服务器上同时运行 50 个微软 Office 会话。因为许可证的限制，在 Windows XP 上这个好功能只能用于一个会话。如果您从远程的机器登录到系统里，那么本地就会退出登录。

如果您安装了一台 Windows Server，可以启动 Terminal Server（终端服务器）服务，使用开放源代码的 **rdesktop** 程序（www.rdesktop.org）从 Linux 工作站访问 Windows 桌面（微软也直接提供了用于 Windows、Mac OS 和 Windows CE 的客户端）。和 VNC 一样，RDP 也是无状态的。因此可以重新连上已经断开的会话，或者执行远程支持。

RDP 甚至能把本地的打印机和磁盘资源提供给远程的 Windows 会话。测试这个设置感受一番吧——只是在您获得许可证之前不要部署它。Terminal Server 给连上来的任何用户颁发 30 天的测试许可证，但 30 天之后，它就不允许那个客户机再连接了，除非您购买了合适的终端服务器客户机访问许可证（Terminal Server Client Access License）。

远程访问 Windows 机器的整个概念似乎把微软和大多数其他 Windows 软件提供商的许可证部门都搞糊涂了。许可证的适用条件随着他们尝试确定自己的政策而一变再变，所以在您吊起用户的胃口之前，要就当前价格取得正式的承诺。我们都乐于看到的一种许可证颁发方案（按并发用户数发许可证）似乎没有提示任何人的议事日程。目前的选择不是给每台计算机购买许可证，就是给每个打算访问服务器的用户购买许可证。

26.3 运行 Windows 和类似 Windows 的应用

商业产品 VMware 能让多个操作系统同时在 PC 硬件上运行。VMware 在宿主操作系统之上模拟了完整的虚拟“客体机器”，而宿主操作系统必须是 Linux 或者 Windows。不管是哪一种宿主操作系统，您都可以把大多数兼容 Intel 的操作系统安装到一台 VMware 的虚拟机器里。

VMware 是适合开发或者测试环境使用的一种理想工具。它甚至能让虚拟操作系统连网，这样一来，它们就能彼此进行通信，共享系统的物理以太网连接。

Wine 系统采取了一种不同的方法。Wine 在 Linux 环境下实现了 Windows 的编程 API，能让您直接在 Linux 和 X 上运行 Windows 应用程序。这个自由软件把本来的 Windows API 调用转成 Linux 上对应的 API 调用，因此不必使用任何微软的代码就可以运行 Windows 应用程序。Wine 支持 TCP/IP 连网、串口设备和声音输出。

Bob Amstadt 在 1993 年开始了 Wine 这个项目。从那以后，Wine 已经走过了漫长的道路。大量的 Windows 应用程序在 Wine 里都可毫无问题地运行，其他应用程序要运行则要借助几个技巧。遗憾的是，让应用程序在 Wine 下运行经常没那么简单。codeweavers.com 的聪明人编写了一个商业的安装系统，它能让一些不干活的 Windows 应用程序正确工作。

如果您选择的工具得到了 CodeWeavers 的支持，那就好了。但即便不是这样，也要让应用程序试试——您或许能有个意外惊喜。可如果应用软件不能自行工作，您也没找到任何重写的提示，而只要您决定自己动手，就要准备好好花些时间把它改出来。如果您有购买的预算，那么可以考虑和 CodeWeavers 签合同，由它们帮助您。

Win4Lin 是 NeTraverse 公司提供的一种商业软件，它能替代 Wine。Win4Lin 声称要比 Wine 稳定，而且支持更多的微软应用。不过，它要求对内核做修改，而 Wine 不需要。

26.3.1 双重引导

如果您在一台过去曾经装过 Windows 的机器上安装 Linux，那么您无疑可以选择设置一种双重引

www.TopSage.com

导的配置。现如今，这样的配置可以随心所欲地发挥功能。甚至有可能把 Windows 的分区挂载到 Linux 下，也可以在 Windows 下访问 Linux 的文件系统。

但是且慢！如果您在干实际工作，既要用 Windows 也要用 Linux，那么把双重引导作为一种可能的解决方案就要留心了。双重引导的设置代表了墨菲定理（译者注：美国爱德华空军基地的一个叫墨菲的测试工程师发明了一条著名的定理。墨菲定理的表述如下："如果一件事情有可能向坏的方向发展，就一定会向最坏的方向发展。"）的最差情况：他们总是好像引导进入了错误的操作系统，最微小的任务也往往要重启系统多次。计算机如此便宜，远程访问如此方便，通常没有理由要让您陷入这样无休止的麻烦之中。

26.3.2 另一种选项：OpenOffice.org

几年前，Sun 发布了 StarOffice 的一种开放源代码的版本，StarOffice 是该公司一种类似于微软 Office 的套装软件，这个版本叫做 OpenOffice.org。OpenOffice.org 包括电子表格、字处理程序、演示软件和绘图应用。这些工具能够读写微软 Office 生成的文件。您可以从 openoffice.org 下载这个套装软件。

包括 Windows、Linux、Solaris 和其他 UNIX 版本在内的所有主流平台上都能用 OpenOffice.org。它还有一个用于 Mac OS X 的版本。如果您在找一个能获得商业支持合同的软件，那么可以从 Sun 购买 StarOffice，它本质上和 OpenOffice.org 是一回事，只是拼写检查做得好，还有一个数据库。

26.4 在 Windows 上用命令行工具

许多 Linux 用户在 Windows 系统上工作时最怀念的就是他们所钟爱的 **xterm**。不像任何老的终端应用程序，或者叫做 DOS 的那个让人讨厌的软件，配好的 **xterm** 支持窗口缩放、彩色显示、鼠标控制以及其他所有漂亮的 **xterm** 转义功能。

虽然 Windows 上没有把 **xterm** 移植为一个独立工具（也就是说，不需要 X），但是有个叫做 **rxvt** 的小程序很接近这个要求。它是 Cygwin 系统的组成部分，从 cygwin.com 可以下载到它。如果您安装了 Cygwin 的 X 服务器，那么就可以用上真正的 **xterm** 了。

Cygwin 这个系统采用 GNU 的 GPL（通用公共许可证，General Public License）来发布，它扩展补充了常用的 UNIX 命令，并且带有一个在 Windows 下实现 POSIX API 的移植库。Cygwin 让 UNIX 和 Windows 命令行及文件系统约定相融合的方式考虑得相当妥善，而且特意把 UNIX shell 的许多方便之处引进为 Windows 自带的命令。除了让 Linux 用户有宾至如归的感觉之外，Cygwin 还能轻而易举地让 UNIX 软件在 Windows 下运行。参考 cygwin.com 这个网站了解详情。

MKS Toolkit 是一种商业版的 Cygwin。参考 MKS 的网站 mkssoftware.com 了解更多信息。

现在能在 Windows 上运行的 UNIX 软件愈来愈多，其中包括 Apache、Perl、BIND、PHP、MySQL、Vim、Emacs、Gimp、Wireshark 和 Python。

26.5 Windows 遵守电子邮件和 Web 标准

在理想世界里，每个人都按开放标准进行通信，从而皆大欢喜。但我们所处的并不是个理想世界，许多人指责 Windows 不但掺入了专有协议，而且破坏了 Internet 标准的实现。或许这个提法不完全错，但的确在标准世界里，也有许多领域 Windows 做得不错。电子邮件和 Web 服务就是这些领

www.TopSage.com

域中的两个。

在 Web 没有形成标准的时代，许多公司和组织在以各种方式采纳和扩展 Web，从而让自己取得竞争优势，获得商业上的大发展。微软在浏览器的层面上也投入到这场混战之中，它给自己的 IE 增加了数不清的特殊扩展。不过，在下面的 HTTP 协议层上，Windows 和 Windows 浏览器都保持了相对的平台无关性。

微软提供了它自己的 Web 服务器 IIS，但是以前 IIS 的性能比运行在 Linux 上的 Apache 大幅度落后。除非用户咬定采取 ASP 这样的服务器端技术，否则没有什么非使用 Windows 机器当 Web 服务器不可的理由。不过，作为 Web 客户机，Windows 机器都能很好地接受 UNIX 和 Linux 服务器提供的全部内容。

对于电子邮件，微软推销自己的产品 Exchange Server 作为首选的服务器端技术。说实话，Exchange Server 的能力要超过那些 Internet 标准的邮件系统，尤其当邮件客户机中有运行微软 Outlook 的 Windows 机器时更是如此。但别害怕：Exchange Server 也能在收发邮件时都用 SMTP 协议，它能使用标准的 IMAP 和 POP 协议给 Linux 客户机送邮件。在客户机这一端，Outlook 和它的免费小兄弟 Outlook Express 都能连接到 IMAP 和 POP 服务器上。您可以自由组合。有关 POP 和 IMAP 的更多知识可以参考 18.3.3 节。

26.6 通过 Samba 和 CIFS 共享文件

在 20 世纪 80 年代初，IBM 设计了一个 API，能让同一子网上的计算机通过名字而不是神秘的数字来彼此通信。这个成果称为 NetBIOS（Network Basic Input/Output System，网络基础输入/输出系统）。NetBIOS 与其最初的下层网络传输协议合起来叫做 NetBEUI（NetBIOS 扩展用户接口，Extended User Interface）。NetBIOS API 变得相当流行，各种不同的网络协议，如 IPX、DECNet 和 TCP/IP 都在协议之上采用了它。

微软和英特尔公司在 NetBIOS 上开发了一种文件共享协议，称为“the core protocol”。随后，又改名为 Server Message Block（服务器消息块）协议，或者简称 SMB。SMB 协议最近演变的结果叫做 CIFS（Common Internet File System，公共 Internet 文件系统），实质上是 SMB 的一个版本，它经过整理和调优，适合在广域网上运行。CIFS 是目前 Windows 上文件共享机制的通用标准。

在 Windows 世界里，能通过网络使用文件系统或者目录就称为可以“共享”。在 UNIX 世界里听上去有点儿奇怪，但是我们在讲到 CIFS 文件系统的时候还是按这个习惯叫。

26.6.1 Samba：UNIX 的 CIFS 服务器

Samba 是一种非常流行的软件包，它采用 GNU 公共许可证，在 Linux 主机上实现了 CIFS 的服务器端。它最初是由澳大利亚人 Andrew Tridgell 开发的，他用逆向工程的方法实现了 SMB 协议，并且在 1992 年公布了代码成果。

现如今，Samba 得到了很好的支持，而且处于活跃的开发之中，它的功能在不断扩大。它提供了一种稳定成熟的机制，可以把 Windows 机器集成到 Linux 网络中来。Samba 的真正动人之处则在于您只要在服务器上安装一个软件包就够了，在 Windows 端不需要任何特殊软件[1]。

CIFS 提供 5 种基本服务：

- 文件共享；

[1] 假定 Windows 机器已经按照“Microsoft 连网方式”配置好了。

www.TopSage.com

TopSage.com

- 网络打印；
- 验证和授权；
- 名字解析；
- 服务声明（文件服务器和打印机“浏览”）。

Samba 不仅能通过 CIFS 提供文件服务，而且它还能执行一个 Windows 主域控制器的所有基本功能[2]。Samba 支持一些高级特性，包括 Windows 域登录、Windows 用户特性描述文件的漫游以及 CIFS 假脱机打印。

Samba 的大多数功能都是由两个守护进程实现的：**smbd** 和 **nmbd**。**smbd** 实现文件和打印服务，还有验证和授权功能。**nmbd** 提供 CIFS 其他的主要成分：名字解析和服务声明。

和 NFS 与内核相互深深地交织在一起不同，Samba 不要求对内核做任何修改，完全作为一个用户进程运行。它绑定到了用于 CIFS 请求的套接口，并等待客户端访问资源的请求。一旦请求通过了身份验证，**smbd** 就派生出自身的一个实例，该实例以发出请求的用户的身份运行。因此会遵守所有常规的文件访问权限（包括组权限）。**smbd** 在这上面所添加的唯一一项特殊功能是文件上锁服务，该功能可向客户端 PC 提供它们所习惯的上锁语义。

26.6.2 Samba 的安装

本书所介绍的所有 Linux 发行版本都带 Samba。补丁、文档和其他资源都可以从 www.samba.org 这个网站得到。要确保使用最新的 Samba 软件包，因为过早的版本可能会造成数据丢失或者安全问题。

在所有的系统上，您都需要编辑 **smb.conf** 文件（不是**/etc/samba/smb.conf**，就是**/etc/smb.conf**）来告诉 Samba 它应该以怎样的方式运行。在这个文件中，您可以指定应该被共享的目录和打印机、它们的访问权限以及 Samba 的一般运行参数。Samba 软件包带的 **smb.conf** 文件样板有很好的注释，对于新配置来说一般是个很好的起点。注意，一旦 Samba 开始运行，它就要每隔几秒检查一次配置文件，并且载入您所做的任何修改。

意识到透过网络共享文件或其他资源所带来的安全隐患是非常重要的。对于一个普通的站点来说，您需要做两件事来达到基本的安全水平。

- 明确规定哪些客户机能够通过 Samba 访问共享资源。这部分配置由 **smb.conf** 文件中的 `hosts allow` 子句控制。要确保其中只包含应该包含的 IP 地址（或地址范围）。
- 阻止从单位之外访问服务器。Samba 只对口令验证加密。它没有对数据传输加密。根据保存在 Samba 服务器上的数据性质，您可能想要阻止从单位外部对服务器的访问，从而防止有人意外地下载到没有加密的文件。一般可以在网络防火墙的层次上做到这一点。Samba 使用 UDP 端口的范围是 137～139，而使用的 TCP 端口的范围是 137、139 和 445。

从 Samba 第 3 版开始，就能从 samba.org 获得非常好的文档了。

Samba 的大多数配置选项默认值都相当合理，大多数网站只需要一个小小的配置文件。使用命令 **testparm -v** 可以获得所有 Samba 配置选项及其当前设置值的一份清单。这份清单包括从文件 **smb.conf** 设置的值和默认值。

要避免在 **smb.conf** 这个文件里设置选项，除非它们和默认值不同，而且您能肯定自己设定它们的原因。这种方法的优势在于，当您升级到 Samba 的一个新版本的时候，您的配置会自动采用 Samba 作者所推荐的设定。

如前所述，要确保打开了口令加密的功能：

[2] Samba 和 LDAP 组合起来用也能做备份域控制器。参考第 17 章了解有关 LDAP 的更多知识。

www.TopSage.com

```
encrypt passwords = true
```

这个选项对 Windows 客户机和 Samba 服务器之间交换的口令进行加密。现在这是默认配置，但是想不出有什么场合您要把它关闭。

加密功能要求 Samba 服务器给每个用户保存一个专门的 Windows 口令散列值。Windows 口令的工作方式和 UNIX 口令的方式根本不同，因此不可能用**/etc/shadow** 里的口令。

Samba 提供了一个专门工具 **smbpasswd** 来设置这些口令。例如，加入用户 tobi，给他设一个口令：

```
$ sudo smbpasswd -a tobi
New SMB password: <password>
Retype new SMB password: <password>
```

用户也可以用 **smbpasswd** 改变自己的口令：

```
$ smbpasswd -r smbserver -U tobi
New SMB password: <password>
Retype new SMB password: <password>
```

这个例子修改了用户 tobi 在服务器 smbserver 上的口令。

26.6.3 文件名编码

Samba 从 3.0 版开始对所有的文件名采用 UTF-8 编码。如果服务器的 LANG 设为 UTF-8，那么会匹配得很好[3]。如果您在欧洲，而且还在服务器上让 LANG 为某种 ISO 8859 模式，那么就会发现，当您在一个目录下输入 ls 命令，而这个目录下的文件是由使用 UTF-8 的 Samba 所创建的时候，像 ä、ö、ü、é 或者 è 这样的特殊字符看上去都变得很奇怪。解决办法是告诉 Samba 使用和您的服务器一样的字符编码：

```
unix charset = ISO8859-15
display charset = ISO8859-15
```

从一开始就保持正确的文件名编码非常重要。不然的话，编码不正确的文件名会越来越多，以后再纠正相当繁琐。

26.6.4 网络邻居浏览

早期的 Windows 连网功能相当简单。只要把所有的 Windows 主机接到一个以太网段，这些系统就开始通过名字互相找到对方，然后能共享资源。这项功能是通过一种基于广播和推选的自组织系统实现的，一台 Windows 主机被选出来成为主管，以后它便维护着一份所有资源的清单。

当 Windows 开始使用 TCP/IP 的时候，这个系统转而使用 UDP 广播。效果相同，只是操作单元是 IP 子网而不是以太网。被找到的主机出现在 Windows 桌面的“网络邻居”或者“我的网络”下，具体取决于特定的 Windows 版本。为了组织得更好看些，每台 Windows 主机都能分配给一个工作组。

Samba 通过守护进程 **nmdb** 加入到这个系统里。**nmdb** 甚至可以变成专门的浏览主管机，履行组织本地子网所有 Windows 系统的责任。遗憾的是，一旦网络开始变得复杂起来，出现了几个子网和服务器，那么只靠广播就不能让所有的 Windows 主机找到对方了。

[3] 键入 **echo $LANG** 查看您的系统是否按 UTF-8 模式在运行。

www.TopSage.com

为了解决这个问题，可以通过在文件 **smb.conf** 里设置：

```
wins support = yes
```

在每个子网里把一台 Samba 服务器配置为 WINS（Windows Internet Naming Service）服务器。

根据我们的经验，微软的浏览系统则好坏无常，甚至是纯粹的 Windows 网络也不行。我们经常会碰到奇怪的情形，比如一台主机对它自己的名字都不响应，或者找不到一台附近的计算机，即便我们能肯定它确实打开了，而且正在运行也不行。对于有多个子网的大型网络来说问题更多。

如果您要向现代的 Windows 客户机（Windows 2000 以后）提供服务，那么要给所有主机都配 DNS 域名，并且不要浏览“网络邻居”，这样一来，如果您愿意，就可以从 Samba 服务器中去掉对 NetBIOS 的支持。

```
[global]
disable netbios = yes
name resolve order = host
```

没有 NetBIOS，Windows 就必须使用 DNS 获得 Samba 服务器的 IP 地址，就像找到网上其他资源那样。如果您运行的 Samba 不支持 NetBIOS，就不需要启动 **nmbd**，可以从**/etc/init** 下的 Samba 启动脚本里把它去掉。

要把一台客户机连到不支持 NetBIOS 的 Samba 服务器上，最好在连接共享资源的时候输入服务器的完整 DNS 域名。例如：

```
\\server.example.com.\myshare
```

注意，服务器名字末尾有个点。它告诉 Windows 不要自找麻烦用 NetBIOS 去找被访问的机器，而是直接用 DNS。

26.6.5 用户身份验证

在 Windows 的身份验证系统中，客户机不信任服务器，用户的口令绝不会以明文形式在网络上传送。相反，Windows 使用一种质询/应答的身份验证方法。

如果您在登录到一个 Windows 系统时提供了用户名和口令，任何时候，只要有验证身份的要求，Windows 就会用这一信息做验证。因为采用了质询/应答的方法，所以除了明文口令保存在内存中的某个地方之外，这样做没有别的危险。如果用户的 Windows 用户名和口令与 Samba 服务器上的正好相同，那么 Samba 能让用户似乎不用口令就可以访问共享资源。所有的身份验证工作都在后台里暗地里做完了。

质询/应答的方法存在缺点，服务器不得不保存等同于明文的口令。事实上，服务器保存的口令副本在本地加密，但这主要是为了防止被意外看到。入侵者如果能接触到加密后的口令，就可以不用破解口令，直接用它们访问相关的账号。Samba 口令必须得到比**/etc/shadow** 文件还要严格的保护。

在采用多个 Samba 服务器的复杂环境下，运行中央目录服务，让同一口令在所有的服务器上都能用，是很有意义的做法。Samba 支持 NIS[4]、LDAP 和 Windows 作为验证服务。NIS 和 LDAP 已经在第 17 章讨论了。

您可以有两种基本的选择，把 Windows 和 Linux 的身份验证系统融合起来。首先，可以配置一台 Samba 服务器充当 Windows NT 4 的主域控制器（不是 Active Directory 服务器）。再就是，可以在

[4] 即使有些文档里还把 NIS+列出来了，但是 Samba 第 3 版不支持 NIS+。似乎是没有人愿意维护代码了，所以这个功能才被去掉（至少现在是这样）。

www.TopSage.com

Windows 客户机上安装 pGina。这个聪明的应用把标准的 Windows 登录系统替换为一种支持所有标准身份验证服务（包括 LDAP 和 NIS）方式的框架。

26.6.6 基本的文件共享

如果每个用户有一个主目录，这些主目录就可以成批共享：

```
[homes]
comment = Home Directories
browseable = no
valid users = %S
writeable = yes
guest ok = no
```

例如，这个配置能让用户 oetiker 从任何 Windows 系统通过路径\\sambaserver\oetiker 访问他自己的主目录。

在有些地方，Linux 主目录的默认权限允许人们浏览别人的文件。因为 Samba 要靠 UNIX 文件权限来实现访问限制，所以从 CIFS 加入的 Windows 用户也能彼此读取别人的主目录。不过，经验表明，这种现象会让 Windows 用户产生错觉，误以为他们的目录暴露在外。前面的配置片段中，valid users 一行告诉 Samba 不许连接其他用户的主目录。如果您不想这么做，就不要这一行。

Samba 使用其神奇的 `[homes]` 作为最后的配置。如果在配置中明确规定某个用户的主目录可以共享，那么在那里设置的参数就会覆盖通过 `[homes]` 设置的值。

26.6.7 用户组共享

Samba 能够把 Windows 的 ACL（访问控制列表，access control list）映射成文件的权限或者 ACL（如果下层的文件系统支持它们的话）。在实际中，我们发现 ACL 的概念对大多数用户来说都显得太过复杂。因此，我们一般只给每组需要共享的用户设置一个专门的共享，配置 Samba 去实现适当的权限设定。只要有一个用户企图挂载那个共享，Samba 就检查看申请者是否在正确的 Linux 用户组内，然后将其有效 UID 换为组共享专门的属主（为此目的而建的伪用户）。例如：

```
[eng]
comment = Group Share for engineering
; Everybody who is in the eng group may access this share.
; People will have to log in using their Samba account.
valid users = @eng
; We have created a special user account called "eng". All files
; written in this directory will belong to this account as
; well as to the eng group.
force user = eng
force group = eng
path = /home/eng
; Disable NT Acls as we do not use them here.
nt acl support = no

; Make sure that all files have sensible permissions.
create mask = 0660
force create mask = 0660
security mask = 0000
directory mask = 2770
force directory mask = 2770
```

www.TopSage.com

```
directory security mask = 0000

; Normal share parameters
browseable = no
writeable = yes
guest ok = no
```

通过 Samba 的 `inherit permissions` 选项也能取得类似的效果。如果在一个共享上启用了该选项，那么所有的新文件和目录都会从其上级目录继承这个设定：

```
[eng]
comment = Group Share for engineering
path = /home/eng
nt acl support = no
browseable = no
writeable = yes
inherit permissions = yes
```

因为 Samba 现在从父目录向下传递设定，所以正确设置好共享的根目录则至关重要：

```
$ sudo chmod u=rw,g=rws,o= /home/eng
$ sudo chgrp eng /home/eng
$ sudo chown eng /home/eng
```

注意，这里的配置仍然要求您创建一个伪用户 eng 作为共享的属主。

26.6.8 用 MS DFS 做透明重定向

微软的 DFS（Distributed File System，分布式文件系统）能让一个共享下的各个目录在客户机需要访问其他共享的时候，触发客户机透明地自动挂载这些共享。对于习惯于 UNIX 和 Linux 的人来说，这个功能听上去没什么大不了，但对于 Windows 来说，整个概念则相当有革命性和出人意料。这也是即便从 Windows 98 起每个版本的 Windows 都支持 DFS，但它仍没有得到广泛使用的部分原因。

下面是一个例子：

```
[global]
; Enable MS DFS support for this Samba server.
host msdfs = yes
...
[mydfs]
; This line tells Samba that it has to look out for
; DFS symlinks in the directory of this share.
msdfs root = yes
path = /home/dfs/mydfs
```

您可以在**/home/dfs/mydfs** 中创建符号链接作为实际的自动挂载点。例如，下面的命令让 **jump** “目录”是到另一台服务器上两个目录中一个的链接（注意用单引号。要用单引号是为了保护反斜线不会被转义）。

```
$ sudo ln -s 'msdfs:serverX\shareX,serverY\shareY' jump
```

如果有不止一个来源（就像本例这样），Windows 会先试一个，不成功再试另一个。访问 **\\server\mydfs\jump** 的用户现在实际上读取的是 serverX 服务器上的 shareX 或者 serverY 上的 shareY，具体取决于哪个可以访问到。如果文件系统按读写方式导出，那么必须保证存在某种机制保持文件的同步。**rsync** 有助于达到此目的。

www.TopSage.com

大家网 TopSage.com

有了 Samba，还可以把访问某个特殊共享的所有客户机重定向到一台不同的服务器上，

```
[myredirect]
msdfs root = yes
msdfs proxy = \\serverZ\shareZ
```

注意，DFS 只适用于在所有相关服务器上的用户名和口令都相同的用户。

26.6.9 smbclient：简单的 CIFS 客户端

Samba 除了在服务器端有许多特性之外，该软件包还包括一个简单的命令行文件传输程序，叫做 **smbclient**。您可以使用这个程序直接访问任何 Windows 或者 Samba 服务器。例如：

```
$ smbclient //redmond/joes -U joe
Password: <password>
Doman=[REDMOND] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]
smb: \>
```

当您成功地登录到文件服务器之后，就可以使用标准的 **ftp** 风格的命令（比如 **get**、**put**、**cd**、**lcd** 和 **dir**）浏览和下载文件。

26.6.10 smbfs 文件系统

Linux 包含对 SMB/CIFS 文件系统客户端的直接支持。您可以把一个 CIFS 共享安装到您自己的文件系统树中，就像您可以把其他任何内核能直接支持的文件系统安装到系统中一样。例如：

```
# mount -t smbfs -o username=joe //redmond/joes /home/joe/mnt
```

虽然这个功能很有用，但要记住一点，Windows 在概念上把网络安装的文件系统看作由一个特殊用户建立（也就是上面的 **username=joe** 这个选项），而 Linux 把它们看作和系统整个融为一体。Windows 服务器一般不支持几个不同的用户同时访问一个安装好的 Windows 共享。

从客户机的角度来看，被安装的目录下所有的目录都属于安装这个目录的用户。如果您以 root 身份安装了共享目录，那么所有的文件都属于 root，普通用户都不能写 Windows 服务器上的文件。

安装选项 **uid**、**gid**、**fmask** 和 **dmask** 能让您调整这些设定，使得文件的所有权和访问权都符合该共享目录专门的访问策略。查看 **mount.smbfs** 的手册页了解这方面的更多知识。

要让用户能自行安装一个 Windows 共享，可以在 **/etc/fstab** 文件中以如下格式加入一行：

```
//redmond/joes /home/joe/mnt smbfs
    username=joe,fmask=600,dmask=700,user,noauto 0 0
```

因为这里给出了 user 选项，所以现在用户能运行下面的命令来安装文件系统：

```
$ mount /home/joe/mnt
```

mount 提示用户在安装共享之前先提供口令。

虽然 NFS 是 UNIX 上网络文件服务的标准，但在有些场合下，使用 Samba 和 CIFS 在 UNIX 和 Linux 计算机之间共享文件更有意义。例如，让用户从个人的笔记本上通过 NFS 安装公司的文件系统有危险[5]。但是，您可以安全地用 CIFS 让这些笔记本能访问他们自己的主目录。参考第 16 章了解有关

[5] NFS 的安全基于这样的思想，即用户在客户机上没有 root 访问权，且客户机上的 UID 和服务器上的 UID 相同。对于自行管理的计算机来说一般都不符合这样的条件。

www.TopSage.com

NFS 的更多知识。

IBM 已经投入研发工作让 Linux 能尽可能无缝地访问 CIFS，并以将 Samba 发展成为代替 NFS 的一种健壮系统为目标。这方面工作的初步成果是 Samba 3.x 能配置成提供“UNIX 扩展”。打开这个功能之后，在通过 Samba 安装一个共享的时候，也能看到全部的 UNIX 文件属性、所有关系和文件类型。

伴随在服务器端的改动，同时还在开发一种客户端文件系统模块，称为 cifs。目前的开发集中在 cifs 模块上，它已经在调试和配置支持上比传统的 **smbfs** 要好了。

26.7 用 samba 共享打印机

共享打印机的简单方法是在 **smb.conf** 文件里加一个 `[printers]` 配置，这就让 Samba 把所有的本地打印机都共享出来。Samba 使用系统的打印命令来完成自己的工作，但由于 Linux 的打印机制不是非常标准，您必须通过把 printing 选项设为一个合适的值，告诉 Samba 您的服务器上使用的是哪一种特殊的打印系统。查看 **smb.conf** 的手册页了解目前支持的打印系统都有哪些。

```
[printers]
; Where to store print files before passing them to the printing system?
path = /var/tmp
; Everybody can use the printers.
guest ok = yes
; Let Samba know this share is a printer.
printable = yes
; Show the printers to everyone looking.
browseable = yes
; Tell samba what flavor of printing system the system is using.
printing = LPRNG
```

Windows 客户机现在能按网络打印机的方式来使用这些打印机了，就好像它们都连接在 Windows 的服务器上一样。不过还有一个小问题。Windows 客户机想知道它正在用的打印机类型，它会请用户选择一种适合的打印机驱动程序。对于在这里不知如何往下进行的用户来说，这就导致他们会提出许多需要支持的请求。如果正好这个打印机需要的驱动程序 Windows 又不带，那么此时就需要更多的支持了。参考第 23 章了解有关打印的知识。

还好，可以配置 Samba 向 Windows 客户机提供所需的 Windows 打印机驱动程序。但要让这个功能发挥作用，您必须做些准备。首先，在 **smb.conf** 文件里加上适当的 `[global]` 配置，让 Samba 能像一台打印服务器那样运行：

```
[global]
; Who is our printer admin
printer admin = printadm
; The following have the right value by default.
disable spoolss = no
; Don't bother showing it; you cannot add printers anyway
show add printer wizard = no
; Assuming you want everybody to be able to print
guest ok = yes
browseable = no
```

现在 Samba 知道它是一台打印服务器，并把 printadm 用户当作它的打印机管理员。

如果您打算给 Windows 客户机提供打印机驱动程序，必须要有一个地方保存这些驱动程序。通过一个叫做 `[print$]` 的特殊共享就能做到。

www.TopSage.com

```
[print$]
comment = Printer Driver Area
; Place to store the printer drivers
path = /var/lib/samba/printers
browseable = yes
guest ok = yes
read only = yes
; Who can administer the printer driver repository
write list = printadm
```

在您可以开始把打印机驱动程序上载到新的打印服务器之前，必须在系统层面上处理好几处细节。确保有 printadm 这个用户，而且有访问 Samba 的权限。

```
$ sudo useradd printadm
$ sudo smbpasswd -a printadm
```

只有在存在适当的目录结构，而且 printadm 拥有该目录结构的情况下，Samba 才能保存打印机的驱动程序（由 `write list` 选项规定）。

```
$ sudo mkdir -p /var/lib/samba/printers
$ sudo cd /var/lib/samba/printers
$ sudo mkdir W32X86 WIN40
$ sudo chown -R printadm .
```

这里有两种选择：您可以走到 Windows 机器旁，从那里上载打印机驱动程序，也可以使用 Samba 的工具完全从命令行上传。遗憾的是，没有简单的办法能知道为某种特定的驱动程序到底安装的是什么，所以我们建议在大多数情况下还是采用第一种方法。只有您亲自把一个驱动程序重复安装到多台服务器上，才值得去研究安装，掌握如何用命令行工具重复这个过程。

26.7.1 从 Windows 安装打印机驱动程序

要从 Windows 客户机安装驱动程序，可以在 Windows 的“开始（Start）”菜单里点“运行（Run）”，在出现的对话框里键入\\samba-server.example.com，连接上 Samba 服务器。Windows 会询问登录到 Samba 服务器的口令。以 printadm 用户身份登录。如果一切顺利，就会弹出一个窗口，列出服务器提供的共享清单。

在子文件夹“打印机（Printers）”下您应该看到服务器共享的所有打印机。用鼠标右键点击打印机图标旁的空白处，弹出“服务器属性（Server Properties）”对话框，然后在“驱动程序（Drivers）”配置页上加上您需要的打印机驱动程序。

上载的驱动程序最后都放在为 `[print$]` 共享指定的目录里。此时您可能想瞥一眼您刚上传的驱动程序的属性。如果您想自动上载驱动程序，就要把这个文件列表提供给 Samba 的命令行工具。

在上传了正确的驱动程序之后，现在就要把它们和特定的打印机联系起来。依次打开每个打印机的属性（用鼠标右键点击并选择“Properties[属性]”），在“高级（Advanced）”配置页里选出合适的驱动程序。然后打开“打印默认设置（Printing Defaults）”对话框，修改设定。即使您乐得使用默认设定，也至少要做一处小改动，让 Windows 把配置数据结构保存在 Samba 服务器上。以后 Samba 就能把数据提供给访问这台打印机的客户机。如果您忘了做这最后一步，可能会引起客户机崩溃，因为它们在试图使用该打印机的时候，没有找到默认的有效配置。

26.7.2 从命令行安装打印机驱动程序

您可能已经猜到，不使用 Windows，其中有些步骤很难重复，特别是设置打印机的默认设定。但

www.TopSage.com

如果您想要在 Samba 服务器上配数百台打印机，那么您一定想从命令行来做。对于 PostScript 打印机来说，命令行的配置效果特别好，因为 Windows PostScript 打印机驱动程序不用默认的配置信息也能正确工作。

如果您记下了一个特定驱动程序所需的文件，那么就可以从命令行安装驱动程序。首先，把所需文件拷贝到[print$]共享里：

```
$ cd ~/mydriver
$ smbclient -U printadm '//samba-server/print$' -c 'mput *.*'
```

接下来，把驱动程序分配给一台特定的打印机。假定您有一台带有自定义 PPD 文件的 PostScript 打印机：

```
$ rpcclient -U printadm -c "\
adddriver \"Windows NT x86\" \"Our Custom PS:\
PSCRIPT5.DLL:CUSTOM.PPD:PS5UI.DLL:PSCIPT.HLP:NULL:NULL:PSCRIPT.NTF\"" \
samba-server
```

为清楚起见，可以把一行分隔为多行，各行以反斜线结尾。如果愿意，您也可以不这么做，把命令写在一行里。双引号之前的反斜线标明嵌套在一对引号里的另一对引号。上例中的长字符串包含打印机驱动程序的属性对话框中列出的信息，在 Windows 安装打印机的驱动程序时能看到这些信息：

- 长长的打印机名字；
- 驱动程序文件名；
- 数据文件名；
- 配置文件名；
- 帮助文件名；
- 语言监视器名（如果没有的话设为 NULL）；
- 默认数据类型（如果没有的话设为 NULL）；
- 由逗号分隔的其他文件列表。

要配置一台打印机使用上传的一个驱动程序，执行：

```
$ rpcclient -U printadm -c "\
set driver \"myprinter\" \"Our Custom PS\"" samba-server
```

26.8 Samba 的调试

Samba 的运行通常不要求多去关照。不过，如果您遇到了问题，可以参考两种主要的调试信息来源：每个客户机的日志文件以及 **smbstatus** 命令。要保证配置文件里对日志文件做了正确的设定：

```
[global]
; The %m causes a separate file to be written for each client.
log file = /var/log/samba.log.%m
max log size = 1000
; How much info to log. You can also specify log levels for components
; of the system (here, 3 generally, but level 10 for authentication).
log level = 3 auth:10
```

日志级别越高，记录的信息越多。记录日志要花时间，所以除非您正在调试问题，否则不要要求记录过多细节。否则运行速度会大大降低。

下面的例子显示了一次不成功的连接尝试之后又有一次成功的连接所生成的日志记录：

www.TopSage.com

```
[2004/09/05 16:29:45, 2] auth/auth.c:check_ntlm_password(312)
  check_ntlm_password:  Authentication for user [oetiker] -> [oetiker] FAILED
     with error NT_STATUS_WRONG_PASSWORD
[2004/09/05 16:29:45, 2] smbd/server.c:exit_server(571)
  Closing connections
[2004/09/05 16:29:57, 2] auth/auth.c:check_ntlm_password(305)
  check_ntlm_password:  authentication for user [oetiker] -> [oetiker] ->
     [oetiker] succeeded
[2004/09/05 16:29:57, 1] smbd/service.c:make_connection_snum(648)
  etsuko (127.0.0.1) connect to service oetiker initially as user oetiker
     (uid=1000, gid=1000) (pid 20492)
[2004/09/05 16:29:58, 1] smbd/service.c:close_cnum(837)
  etsuko (127.0.0.1) closed connection to service oetiker
[2004/09/05 16:29:58, 2] smbd/server.c:exit_server(571)
  Closing connections
```

不必去修改 **smb.conf** 文件，**smbcontrol** 命令可以方便地改变正在运行的 Samba 服务器上的调试级别。例如：

```
$ sudo smbcontrol smbd debug "4 auth:10"
```

上面的例子把全局的调试级别设为 4，把与身份验证有关的调试级别设为 10。参数 **smbd** 指出系统上所有的 **smbd** 守护进程都设了调试级别。要调试一条已建立的特定连接，可以使用 **smbstatus** 命令判断出该连接在由哪个 **smbd** 守护进程处理连接，并把它的 PID 传给 **smbcontrol**，只对这条连接进行调试。日志级别超过 100 的话，就能在日志中开始看到（加密后的）口令了。

smbstatus 显示了当前活动的连接和被锁住的文件。当您跟踪上锁问题（例如，“哪个用户以读写独占方式打开了 **xyz** 文件？”）时，这一信息特别有用。输出清单中的第一段列出了用户已经连上的资源，第二段列出了任何活动的文件锁[6]。

```
Samba version 3.0.5

PID     Username  Group     Machine
-------------------------------------------------------------------------------
12636   zauck     ee        zhaka (192.168.1.228)
29857   milas     guests    beshil (192.168.1.123)

Service  pid     machine  Connected at
-------------------------------------------------------------------------------
milasa   29857   beshil   Fri Sep  3 17:07:39 2004
zaucker  12636   zhaka    Thu Sep  2 12:35:53 2004

Locked files:

Pid     DenyMode    Access  R/W     Oplock Name
-------------------------------------------------------------------------------
29857   DENY_NONE  0x3      RDWR  NONE  /home/milasa/hello.dba
12636   DENY_NONE  0x2019f  RDWR  NONE  /home/zaucker/aufbest.doc
```

如果您杀死了与某个用户关联的 **smbd**，那么属于它的全部锁都将消失。有些应用处理这个问题更妥善，只有它们需要一个锁的时候才再度获得它。而别的应用（比如微软的 Access）就会停止运行并死掉，而且要在 Windows 端用鼠标点好多下才能把这个不高兴的应用关掉。糟糕的是，我们会看到文件因这样的处理而损坏。无论如何，当 Windows 抱怨说文件被某个别的应用锁住的时候，都要小心。Windows 往往是对的，您应该通过关闭应用，在客户机端改正问题，而不是在服务器上粗暴地处理。

[6] **smbstatus** 的输出信息包括一些很长的行，我们为了清晰起见将其进行了压缩。

www.TopSage.com

TopSage.com

26.9 推荐读物

TERPSTRA, JOHN H. *Samba-3 by Example: Practical Exercises to Successful Deployment (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. （在 samba.org 上有本书的在线版本。）

TERPSTRA, JOHN H., JELMER R. VERNOOIJ. *The Official Samba-3 HOWTO and Reference Guide (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. （在 samba.org 上有本书的在线版本。）

26.10 习题

E26.1 为什么要封锁从 Internet 上对一台 Samba 服务器 137-139 端口以及 445 端口的访问？

E26.2 在一台 Windows 机器上安装 Cygwin 软件，在 **rxvt** 中使用 **ssh** 连接到一台 Linux 机器上。您发现 PuTTY 哪些不同？

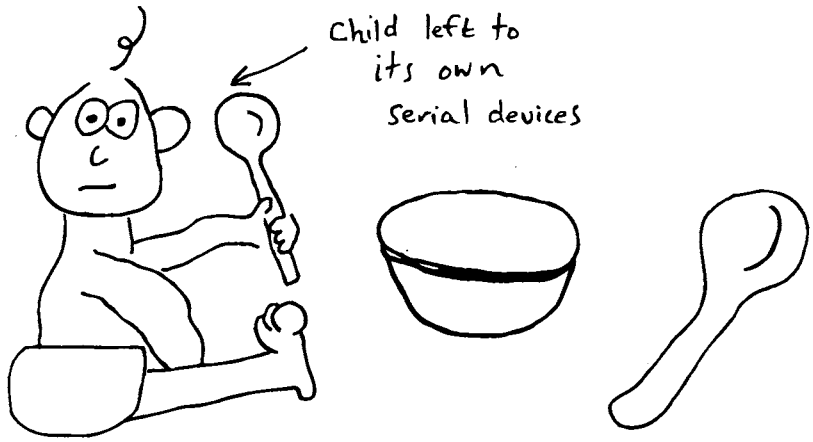
★★E26.3 在实验室里，比较通过 Samba 访问文件的客户机与从 CIFS 原本的服务器（也就是说，Windows 机器）访问文件的客户机的性能。如果两台测试用的服务器性能不同，则要设计一种方法，调整硬件的不同，以便让比较更能体现出服务器软件的性能来。（可能要求有 root 访问权限。）

★★E26.4 在实验室里，使用一个包探测器（**tcpdump**）或者 Wireshark，监视 Windows 和 Linux 之间的一次 telnet 会话。下载并安装 PuTTY 软件，重复上述监视过程。在每种情况下，您从包探测器能看到什么？（要求有 root 访问权限。）

★★E26.5 配置一台 Samba 打印机服务器，提供它所共享的所有打印机的驱动程序。保证各个打印机都有合理的默认配置。

www.TopSage.com

第 27 章　串行设备



从 1969 年起，RS-232C 串行接口标准就为许多厂商制造的许多种设备提供了一条通信途径。它是一种在几乎 40 年历史中实际上保持不动的计算机接口标准，而且如今它还在使用。

串口可以在很多不同的设备中使用，包括打印机、终端和其他的计算机。它们还可以在很多定制的、业余爱好者制作的和数据量少的设备中找到（磁带更换机、温度传感器、GPS 接收器、甚至缝纫机中）。一个串行设备可以直接连接（使用一根电缆）到系统上，也可以通过两端有调制解调器的电话线连接到系统上。

本章讲述了如何将串行设备连接到系统中，并解释了如何配置软件去使用它们。我们会经常使用调制解调器以及打印机作为特殊例子，但其他设备本质上和它们类似。

前几节将讲述串行硬件设备和电缆连接需要考虑的事项。然后，我们从第 27.8 节开始探讨历来用以支持硬件终端的软件基础结构。很少有人还在使用终端，但 Linux 处理伪终端和窗口系统时，依然能够看到它们的影子。本章其余的部分（从 27.14 节开始）提供了关于调制解调器、串行调试以及 USB（通用串行总线，Universal Serial Bus）的一些背景知识。

27.1　串行标准

大多数串口都遵循 RS-232C 标准的某个变种。这个标准规定了每一条信号线的电气特性和含义，以及图 27.1 中所示的传统 25 针（DB-25）串行连接器中的针脚/针孔分配。

www.TopSage.com

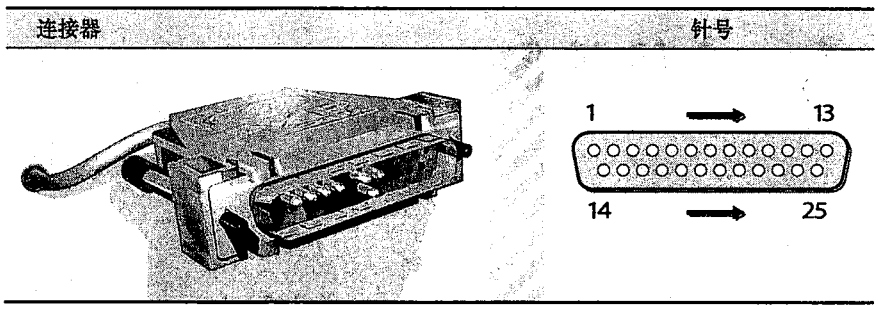


图 27.1 针式 DB-25 连接器

RS-232C[1]标准的全部功能超出了现实世界所有情况下的需要，因为它定义了许多对于基本通信而言没有必要的信号。DB-25 连接器也太大了，用起来不方便。结果，有些替代连接器得以广泛应用。这些连接器将在 27.2 节中介绍。

传统的 RS-232 使用屏蔽双绞线电缆（shield twisted-pair cable，STP），通常为 22 号多股导线。现在串行连线使用无屏蔽双绞线（unshielded twisted pair，UTP）和以太网连接所采用的标准 UTP 一样。这种线在技术上并不符合 RS-232 规范，但通常在实际中用起来没问题。

RS-232 信号电压最初是±12V 直流电压，现在更常用±5V 直流电压。有些时候也使用±3V。电压越高越不易受到干扰。所有这些电压值都符合 RS-232 规范，所以把使用不同电压标准的设备连接起来是完全没有问题的。

RS-232 不是电气“平衡”系统。在每个方向上，它都使用一根单独的导线用于数据传输。因此，对于串行通信来说，双绞线的特殊电气特性并没有像对以太网那样重要。事实上，如果将两根数据线（TD 和 RD）一起放在一对线中，那么使用双绞线电缆就会降低串行电缆的可靠性和使用范围。所以不要这么做。

对于哪些 RS-232 信号应该在一对双绞线电缆中一起传输，并没有意见统一的标准。有些资料推荐使信号地与 TD 和 RD 分别成对，但这样结对的话，要多占一条导线，而且有了到信号地的多条途径。据我们所知，没有非要使用此规则不可的理由。

DB-25 连接器是针式的（上面有伸出的针，叫做 DB-25P）或者孔式的（上面有和针匹配的孔，叫做 DB-25S）。在针或孔附近有一些小到几乎看不见的数字，从 1 到 25 标明了它们的序号。您只有把连接器对着光线，并且从一定的角度看它，才能够清楚地看到这些数字。有时候，只有 1、13、14、25 这几根针标了数字。

图 27.1 展示了一个针式的 DB-25 连接器。和所有的串行连接器一样，孔式连接器上孔的标号是针式连接器上针的标号的镜像。该表格根据图示的方向绘制（假设您正在面对电缆的端部，连接器将插向您的面前）。

请注意，在图 27.1 中，实际上只安装了 7 根针。这是典型的实际使用情况。RS-232 信号和它们在 DB-25 连接器上分配的针如表 27.1 所示。只有阴影部分的信号才在实际中使用（至少在普通的计算机系统中是这样），其他所有信号都可以被忽略。

表 27.1 DB-25 连接器中的 RS-232 信号和针分配

针	名 称	功 能	针	名 称	功 能
1	FG	保护接地	3	RD	接收数据
2	TD	传输数据	4	RTS	请求发送

[1] 为了保持技术上的正确，这个标准现在应该称为 EIA-232-E。然而，没有人会对您所谈论的这些有一点儿概念。

www.TopSage.com

续表

针	名 称	功 能	针	名 称	功 能
5	CTS	允许发送	16	SRD	辅助 RD
6	DSR	数据设置就绪	17	RC	接收时钟
7	SG	信号地	18	-	没有指定
8	DCD	数据载波检测	19	SRTS	辅助 RTS
9	-	正电压	20	DTR	数据终端就绪
10	-	负电压	21	SQ	信号质量检测
11	-	没有指定	22	RI	振铃指示
12	SDCD	辅助 DCD	23	DRS	数据速率选择
13	SCTS	辅助 CTS	24	SCTE	外部时钟传输
14	STD	辅助 TD	25	BUSY	忙
15	TC	传输时钟			

对于串行设备而言有两种接口配置：DTE（Data Terminal Equipment，数据终端设备）和 DCE（Data Communications Equipment，数据通信设备）。DTE 和 DCE 使用相同的插脚引线，但它们对 RS-232 信号的解释不同。

每一个设备要么被配置成 DTE，要么被配置成 DCE。只有少数设备对这两种方式都支持，但不能够同时使用。计算机、终端和打印机通常情况下是 DTE，大多数调制解调器是 DCE。DTE 串口和 DCE 串口能以任意组合进行通信，但不同的组合需要不同的接线方式。

DTE 和 DCE 并存并没有什么合理的理由，所有设备都能够使用同一种连线方案。这两种约定的存在纯粹是 RS-232 许多无意义的历史遗留之一。

如果过多地考虑 DTE 和 DCE 的含义，那它们就会变得非常让人迷惑。在出现这种情况时，做一个深呼吸，然后重新阅读以下要点。

- 对于给定的连接器类型而言，RS-232 插脚引线总是相同的，不管这个连接器是针式的还是孔式的（匹配的针编号就能够连接），也不管这个连接器是在电缆、DTE 设备或者 DCE 设备上。
- 所有的 RS-232 术语都是以从 DTE 设备到 DCE 设备直通连接的模型为基础的（使用“直通连接”一词，我们的意思是说 DTE 端的 TD 针/孔连接到 DCE 端的 TD 孔/针，以此类推。每个针/孔连接到另一端相同编号的孔/针）。
- 信号是从 DTE 设备的角度进行命名的。例如，TD（transmitted data，传输数据）实际的意思是“数据从 DTE 传输到 DCE”。尽管名字叫做传输，但是 TD 针是 DCE 设备上的一个输入针。与此类似，对于 DTE 而言，RD 是输入针，而对于 DCE 而言，它是输出针。
- 把 DTE 设备连线到 DTE 设备时（计算机到终端或者计算机到计算机），必须哄骗每个设备把另一个设备当作是 DCE。例如，两个 DTE 设备都希望在 TD 针上进行传输，在 RD 针上进行接收，您必须交叉连接导线，这样的话，一个设备的传输针就和另一台设备的接收针相连，反之亦然。
- 对于 DTE 到 DTE 的通信，有 3 组信号必须按这种方式进行交叉连接（如果选择全部连接它们的话）。TD 和 RD 必须进行交叉。RTS 和 CTS 也必须进行交叉。每一端的 DTR 针都必须和对端的 DCD 以及 DSR 针相连接。
- DTE 到 DTE 通信电缆的交叉连接常称为“空调制解调器（null modem）”电缆，这又加剧了混乱。您可能会尝试着使用一个空调制解调器电缆来挂接一个调制解调器，但因为调制解调器是 DCE 设备，所以根本就不起作用！调制解调器的电缆被称为是“调制解调器电缆（modem cable）”或者“直通电缆（straight cable）”。

www.TopSage.com

因为 DTE 对 DCE 的问题这么容易造成混淆，所以，您可能会偶尔碰到那些出于好意但却是错误的建议：通过定义 DTE 和 DCE，就好像它们有各自单独的插脚引线似的（比如，将 DCE 的 TD 针重命名为 RD，反之亦然），来弄清楚这个命名方案。在这个替代方案中，插脚引线改变了，但是电缆连接（通过信号名称指定）却是相同的。我们建议您不要理睬那些谈论“DTE 插脚引线”或者“DCE 插脚引线”的任何资料，因为它不太可能是可靠的信息来源。

原本应该是 DTE 设备使用针式连接器而 DCE 设备使用孔式连接器。最后，硬件的设计者意识到针式连接器更容易损坏。现在那些昂贵的计算硬件设备通常使用孔式的连接器，大多数电缆的两端都是针式连接器[2]。

图 27.2 展示了空调制解调器电缆和直通电缆中的针的布局和连接关系。图中只给出了实际用到的信号。

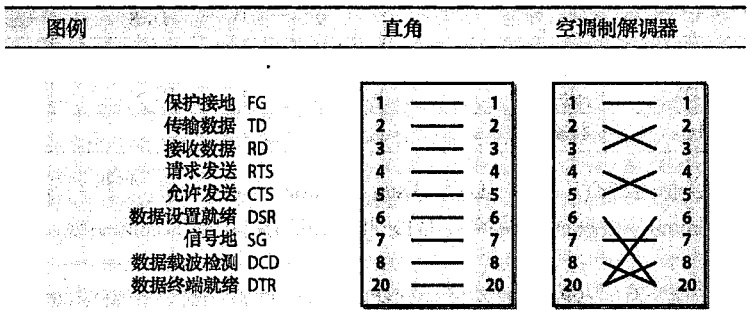


图 27.2 DB-25 电缆的针的布局和连接

27.2 替代连接器

下面几小节讲述那些最常见的替代连接器：微型 DIN-8、DB-9 和 RJ-45。尽管它们在物理上有所不同，但这些连接器都能够接上和 DB-25 相同的电气信号。如果使用正确类型的转换器电缆，那么不同连接器的设备一定能够互相兼容。

27.2.1 微型 DIN-8 连接器

在许多便携机和工作站上都可以找到微型 DIN-8 连接器。这个几乎呈圆形并且极为紧凑的连接器提供了 7 种信号的连接。在图 27.3 中对它进行了说明。

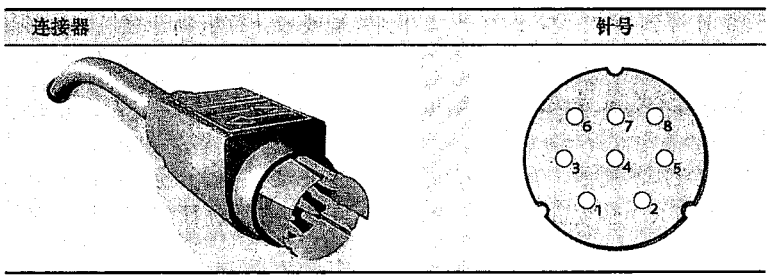


图 27.3 针式微型 DIN-8 连接器

[2] “male（公，针式）”和“female（母，孔式）”这两个词不太合适。现在鼓励大家使用“plug（插头）”和“receptacle（插座）”这两个词。标准的连接器名 DB25P 和 DB25S 实际上源自于第 3 种习惯称谓：“pin（针）”和“socket（口）”。

www.TopSage.com

街上的计算机零售商通常能提供预制的 DB-25 到微型 DIN-8 的转换器电缆。千万不要试着自己去做这些东西，因为一个微型 DIN-8 连接器相当小，用手指很难保证其连接的可靠性。微型 DIN-8 连接器上针的分配如表 27.2 所示。

表 27.2 微型 DIN-8 到 DB-25 直通电缆的针对照

DIN-8	DB-25	信 号	功 能
3	2	TD	传输数据
5	3	RD	接收数据
6	4	RTS	请求发送
2	5	CTS	允许发送
4、8	7	SG	信号地
7	8	DCD	数据载波检测
1	20	DTR	数据终端就绪

27.2.2 DB-9 连接器

这种 9 针连接器在 PC 中很常见（它看起来像“小一号”的 DB-25），它可以提供 8 种最常用的信号。

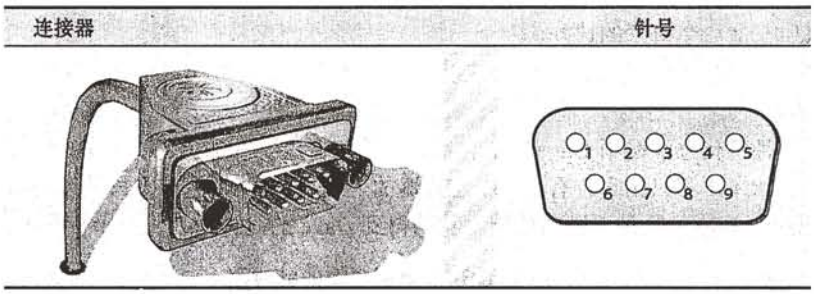


图 27.4 针式 DB-9 连接器

您所在地区的 PC 零售商会提供预制的 DB-9 到 DB-25 的转换电缆。表 27.3 给出了 DB-9 连接器上针的分配情况。

表 27.3 DB-9 到 DB-25 直通电缆的针对照

DB-9	DB-25	信 号	功 能
3	2	TD	传输数据
2	3	RD	接收数据
7	4	RTS	请求发送
8	5	CTS	允许发送
6	6	DSR	数据设置就绪
5	7	SG	信号地
1	8	DCD	数据载波检测
4	20	DTR	数据终端就绪

27.2.3 RJ-45 连接器

RJ-45 是一个模块化的 8 芯电话连接器。它和在美国用于电话连线的标准 RJ-11 连接器非常相似，

www.TopSage.com

但 RJ-45 是 8 芯（而 RJ-11 只有 4 芯[3]）。在许多情况下，使用 RJ-45 会让通过建筑物里已有的线路设施（可能是按双绞线以太网的考虑安装的）进行串行通信的工作更方便。

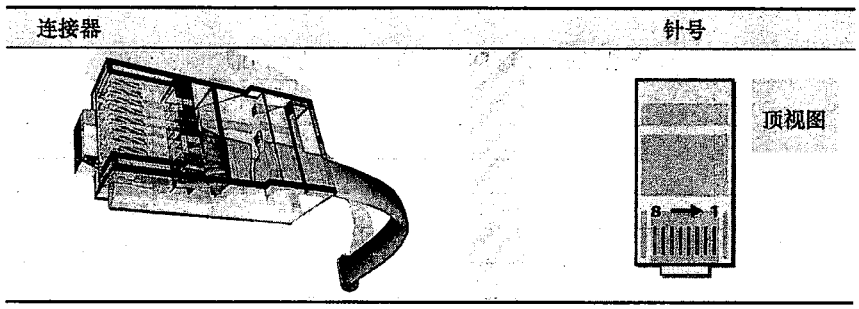


图 27.5 针式 RJ-45 连接器

用于串行连接的 RJ-45 插座在计算机或者普通的串行设备中一般找不到，但是当串行线需要经过接插板时，它们经常用作中间连接器。RJ-45 有时会和扁平电话线，而不是双绞线一起用。虽然 STP 可能在长距离上有更好的信号质量，但对于串行连接而言，这两种电缆都可以用。以太网连接绝不能用扁平电话线，所以许多地方标准采用双绞线电缆，减少无知的用户不小心拿错了电缆。

RJ-45 简洁轻巧，能够自我保护，而且价格便宜。它们通过一种特别的工具被钳接到电缆上去。钳接一个 RJ-45 连 1 分钟的时间都不用。如果您正在从头开始设计一个大规模的布线系统，对于中间连接器而言，RJ-45 是一个很好的选择。

有一些系统可以将 RJ-45 连接器上的针映射到 DB-25 中去。表 27.4 给出了正式的 RS-232D 标准，只有偶尔才会用到。

RJ-45 的另一种替代办法是用 Dave Yost 标准，它给每台设备都加上了一个 RJ-45 插座，不管这个设备是 DTE 还是 DCE，都只用一种有 RJ-45 连接器的电缆线。Dave 标准不只是针的布局，它还是一种完善的布线系统，解决了几种不同的问题。我们得到 Dave 的允许，在下一节介绍 Dave 的标准。

表 27.4 RJ-45 到 DB-25 直通电缆的针对照

RJ-45	DB-25	信 号	功 能
1	6	DSR	数据设置就绪
2	8	DCD	数据载波检测
3	20	DTR	数据终端就绪
4	7	SG	信号地
5	3	RD	接收数据
6	2	TD	传送数据
7	5	CTS	允许发送
8	4	RTS	请求发送

27.2.4 RJ-45 连线的 Yost 标准

这个方案对几个 RS-232 的难题提供了解决办法（这是由 Dave Yost 编写的规范）：

- 所有的电缆连接器都是相同的类型（针式 RJ-45）；
- 在 DTE 和 DCE 之间不做任何区别；

[3] 仔细观察就会发现 RJ-11 插头实际上有 6 个插入针的槽，但是只安装了 4 条导线。真正的 4 线电话连接器是 RJ-10。安装全了 6 线的 RJ-11 则称为 RJ-12。

www.TopSage.com

- 只需要一种连接器电缆；
- 只需使用一个钳子就可以快速地大规模制作电缆。

每个设备上的每一个串口都有它自己的正好能把 DB-25 或者 DB-9 转到 RJ-45 的适配器。这个适配器永久地固定到了端口上。于是端口现在都成了同种的连接器接口，即孔式 RJ-45，而不论它下面的连接器是 DB-25 还是 DB-9，是 DCE 还是 DTE，是针式的还是孔式的。而且每个串口现在都以相同的针传输和接收数据。

一旦把这些适配器安到了 RS-232 端口上，就能在不需要使用空调制解调器或者空终端，也不需要改变电缆连接器上的针布局或是制作特殊电缆的情况下，连接任意两台设备。您可以将调制解调器连接到计算机上、将调制解调器连接到终端上、将终端连接到计算机上、将终端连接到终端上以及将计算机连接到计算机上等，所有这些连接只需使用一种电缆即可。

在 Yost 最初的设想中，这些电缆都是有护套的 8 芯带状电缆。电缆每一端的连接器通过一种钳子给钳到电缆上，因此既不需要焊接针，也不会把针搞混。因为反感这种电缆，有些单位把 Yost 标准用于 UTP 布线。

电缆的每个方向上都有 3 根信号线（一根数据线和两根控制线），除此之外还有一对信号地。这些电缆不是按通常的方式进行连线的（也就是说，连接器上的每根针都和电缆另一端对应的针进行连接）。它们是使用“经过交叉的”、“互为镜像的”、“边对边反转的”，或者任何您想称呼它的方式进行连线的。也就是说，电缆一头的 1 号针和电缆另一头的 8 号针相连（这个“经过交叉”的意思和“双绞线”中的意思是不同的）。这个方案可行，因为带状电缆中的信号布局是对称的。也就是说，每个传输针穿过扁平的电缆[4]，在其镜像导线的位置上都有相应的接收针。

现成的 RJ-45 电缆都是直通的。为了在这样的系统上使用它们，您必须剪掉电缆一端的连接器，然后颠倒导线的顺序钳接一个新的连接器。孔式对孔式的 RJ-45 连接器（“对接单元”）可以用来延长电缆的长度，但必须记住：使用这样一个双口连接器连接两根交叉电缆就构成了一根直通电缆。

许多厂商都生产 DB-25 到 RJ-45 的适配器。它们的内部颜色编码和电缆的颜色不匹配。适配器、导线连接器和导线都可以在电子商店中买到，但可悲的是，没有关于如何把它们用在 RS-232 上的任何帮助。

Yost 方案的目的是为了同带护套的带状电缆配套使用，电缆中所有的导线都是并排的。与此相对照的是，双绞线电缆有 4 对导线，每一对导线都在沿着导线长度的方向进行了交叉。如果使用双绞线电缆（比如说 5 类电缆），就不应该像通常连接 RJ-45 那样连接电缆（比如说，用于 10BaseT、电话等）。更合适的方法是，您应该让导线 3:4 成一对，5:6 成一对。其他的成对方法易受数据信号串扰的影响。剩余导线的成对不是很重要，但 1:2 和 7:8 都应该和其他导线对一样连接好。请参看 14.2.3 节中关于 5 类电缆的更多信息。

在适配器内部有一个 RJ-45 插槽，它引出了 8 根导线。这些导线都接了 RS-232 针（或者插孔，随情况而定）。只需将这些针推入 RS-232 连接器上正确的孔内，把适配器固定住即可。使用表 27.5 所示的针脚接线。

表 27.5 Yost RJ-45 到 DB-25 或者 DB-9 的适配器的连线

RJ-45 电缆		适配器	连接到 DCE 针			连接到 DTE 针		
			DB-25	DB-9	信 号	DB-25	DB-9	信 号
1	棕（到灰）	蓝	4	7	RTS	5	8	CTS
2	蓝（到桔）	桔	20	4	DTR	8	1	DCD

[4] Dave 没有把这个说得很清楚，但实际上必须在没有造成物理上交叉的情况下连接电缆，以获得“交叉”的效果。因为电缆端部的连接器都是彼此相对的，所以它们的引脚编号就自动颠倒过来了。

www.TopSage.com

续表

RJ-45 电缆		适配器	连接到 DCE 针			连接到 DTE 针		
			DB-25	DB-9	信 号	DB-25	DB-9	信 号
3	黄（到黑）	黑	2	3	TD	3	2	RD
4	绿（到红）	红	7	5	GND	7	5	GND
5	红（到绿）	绿	7	5	GND	7	5	GND
6	黑（到黄）	黄	3	2	RD	2	3	TD
7	桔（到蓝）	棕	8	1	DCD	20	4	DTR
8	灰（到棕）	白	5	8	CTS	4	7	RTS

然而，还存在一个问题，那就是：两个信号地的针不得不进入同一个 DB-25 或者 DB-9 的孔中（分别是 7 针或者 5 针）。可以使用 AMP 制作的名为“Tel-splice connector 1/2 tap dry”（部件号为 553017-4）的塑料工具将这两根导线压到一起，这样它们就可以只和一个针相连。迄今为止，这个部件只能以 1000 个为单位批量购买，价格约为 80 美元。请相信我[Dave Yost]，要是您打算连线的适配器不只几个的话，那就用得着它们。

有些 DTE 设备在发送数据之前要求有 DSR 信号。这个信号通常是由 DCE 设备提供的，但您可以通过把 20 针和 6 针连接起来（在 DB-9 连接器中是 4 针和 6 针）去伪造这个信号。通过这种方法，DTE 设备可以在它发出 DTR 信号的任何时候从它自身接收到 DSR 信号。

在有些 DCE 打印机上，RJ-45 适配器中的 7 针（棕色的导线）应该连接到 DSR 线上（DB-25 和 DB-9 中的 6 针）。请阅读打印机的文档来确定该打印机是否是在 DSR 而不是在 DCD 上提供了有用的联络信号。

一定不要把上述的 DSR 窍门当作教科书去用（Dave Yost，加州洛斯拉图斯，1999 年 7 月）；只在看上去需要它们的（或者看上去不能用标准方法的）特定设备上再用。有些“普通”设备能容忍额外的连线，而别的设备则会被弄糊涂。

27.3 硬载波和软载波

当接上某个设备并打开电源时，Linux 希望看到 DCD（即载波检测信号）变为高电位（正电压）。该信号是在标准 DB-25 连接器的第 8 针上传送的。如果串行电缆有了一根 DCD 线，并且计算机确实会关注这个信号，那么您正在使用的就是硬载波。大多数系统也允许使用软载波，在软载波系统中计算机都假装始终有 DCD 信号。

对于某些特定的设备（尤其是终端），软载波是非常有用的。它可以让您给每个串行连接只使用 3 根信号线：传输、接收和信号地。不过，调制解调器连接确实需要 DCD 信号。如果通过调制解调器连接终端，而丢失了载波信号，那么调制解调器就会挂起（尤其是在长距离的拨号上！）。

在正常情况下，您可以在配置文件中（例如，登录终端用**/etc/gettydefs** 或者**/etc/inittab**，打印机用**/etc/printcap**）给正在使用的、同某个串口相关联的客户软件配置串口的软载波。还可以使用 **stty-clocal** 随时启动软载波。例如：

```
# stty -clocal < /dev/ttyS1
```

将在端口 **ttyS1** 上启动软载波。

27.4 硬件流控

信号 CTS 和 RTS 用于确保设备发送数据的速度不快于接收器能够处理的速度。例如，如果调制

www.TopSage.com

解调器正处于用完缓冲空间的危险之中（也许是因为与远程站点的连接比本地计算机和调制解调器两者之间的串行连接速度慢的缘故），它就通知计算机沉默片刻，直到在缓冲区中得到更多的空间。

流控对于高速调制解调器而言是非常重要的，对打印机也非常有用。在那些不支持硬件流控的系统中（要么是因为串口无法理解它，要么是因为串行电缆没有连接 CTS 和 RTS），流控有时候能够由使用 ASCII 字符 XON 和 XOFF 的软件来模拟。然而，软件流控必须得到高级软件的明确支持，而且即使如此它也不一定能工作得很好。

XON 和 XOFF 分别是<Control-Q>和<Control-S>。对于 **emacs** 用户来说，这可是个问题，因为<Control-S>是 **emacs** 搜索命令所绑定的默认键。为了解决这个问题，可以把搜索命令绑定到别的键上，或者使用 **stty start** 和 **stty stop** 命令改变终端驱动程序对 XON 和 XOFF 的理解。

大多数终端都会忽略 CTS 和 RTS 信号。把电缆在终端一端的 4 针和 5 针跳接在一起，就可以骗过那些在进行通信之前要求在这些针上握手的终端。当终端在 4 针上发出一个信号说“我已经准备就绪啦”，它就会在 5 针上得到同样的信号回复说“继续”。您还可以用这样的方式把 DTR/DSR/DCD 握手信号跳接起来。

和软载波一样，可以通过配置文件或者 **stty** 命令来管理硬件流控。

27.5 电缆长度

RS-232 标准规定在 9 600 波特率下最大的电缆长度为 75 英尺。标准通常是很保守的，RS-232 当然也不例外。我们经常以更长距离使用 RS-232 电缆，最长大约达到了 1 000 英尺。我们在 800 英尺到 1 000 英尺之间的某处碰到了长度极限，但我们发现，每一端采用特殊品牌的设备，结果有很大的不同。

有时把导线驱动器或者中继器和 RS-232 一同使用，延长连接的最大长度。遗憾的是，这些设备常常只能延长 RD 和 TD，所以别的信号可能需要跳接。

27.6 串行设备文件

串口是通过**/dev** 中的设备文件来表示的。大多数计算机都有两个内置的串口：**/dev/ttyS0** 和 **/dev/ttyS1**。Linux 的各个发行版本通常为更多的串口预装了补充的设备文件（64 个或者更多），但是除非给系统加上了更多硬件，否则多出来的这些文件都是多余的，应该忽略不去管它们。

和通常一样，设备文件的名字实际上没什么要紧关系。设备的行为是由主次设备号决定的，设备文件的名字仅仅是为了方便人们使用罢了。

27.7 setserial：把串口参数通知给驱动程序

PC 上的串口可以设置使用几种不同的 I/O 端口地址和中断号（IRQ）。这些设定一般在加电启动时通过系统 BIOS 来访问。改动它们最常见的理由是容纳某个对自身的设置过分挑剔的，只有当它已经指定好串口正常采用的设置时才能正确工作的怪僻硬件。遗憾的是，没有您的帮助，串口驱动程序可能不会检测到这样的配置变化。

UNIX 对这样的多变性所采取的办法是在编译内核时指定串口的参数。幸运的是，Linux 能让您跳过这个费力的步骤，用 **setserial** 命令随时改变串口的参数。**setserial -g** 命令显示出了当前的设置：

www.TopSage.com

TopSage.com

```
# setserial -g /dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
```

要设置参数，您可以指定设备文件，然后给出一系列参数和值。例如，命令：

```
# setserial /dev/ttyS1 port 0x02f8 irq 3
```

设置 **ttyS1** 的 I/O 端口地址和 IRQ 值。重要的是要牢记，这条命令绝对不会改变硬件的配置，它只是把配置告诉给 Linux 串口驱动程序。要改变硬件的实际设置，请参考系统的 BIOS。

setserial 只改变当前配置，系统重新启动之后设置不会保留。遗憾的是，没有一种让改动一直有效的标准方法，我们举例的每种 Linux 发行版本处理它的方式都不一样。



Red Hat 和 Fedora 的脚本**/etc/rc.d/rc.sysinit** 检查**/etc/rc.serial** 文件是否存在，如果有这个文件，就在引导系统的时候执行它。因为系统没有提供样板文件，所以如果您要用到这项功能，则必须自己创建这个文件。只要把您想要运行的 **setserial** 命令列在这个文件中，每条命令一行即可。为了完善起见，让文件是可执行的，并且在第一行里加上**#!/bin/sh** 或许是个好主意，但这些并不必要。



SUSE 提供了一个脚本**/etc/init.d/serial**，它负责处理串口的初始化。遗憾的是，这个脚本没有配置文件，您必须直接编辑它来体现您想要执行的命令。糟糕的 SUSE！脚本使用它自己的微型元语言来构建 **setserial** 命令行，但幸运的是，大量注释行给出了可供选择的例子。



Debian 提供了一个配置文件**/etc/serial.conf**，供**/etc/init.d/setserial** 读取，它有很好的文档。它提供了多种高级特性（比如自动配置），可能只对发行版本的设计者有用。在简单的情况下，只要从注释中找到您想要的行，或者以 **setserial** 的命令格式加上您自己的行（省略命令名）即可。

27.8 串行设备的软件配置

一旦用正确的电缆连接好了一个设备，就必须配置主机上的软件以便使用该设备。对新设备所进行的繁琐配置是由设备类型以及它应该发挥的作用决定的。

- 对于直接相连的终端而言，您必须通知系统监听终端端口上的登录消息。您可以指定串行连接的速率和参数。终端的配置将在下一节中讨论。
- 拨入式调制解调器的配置和直接相连的终端相似。然而，在有些系统中，准确的过程可能稍有不同。
- 要了解如何使用调制解调器通过 PPP 协议连接一个远程网络，请参考 12.13 节中的内容。
- 请参考第 23 章，了解设置串行打印机的相关信息。有些打印机只能接收数据，其他打印机则是双向传输的，能把它的状态信息返回给主计算机。
- 只从自己的软件中使用的自定义串行设备不需要专门配置。只需打开设备文件就能访问该设备。请参考 **termios** 的手册页，了解如何设置串口的速率、标志位和缓冲模式。

27.9 硬件终端的配置

在过去的 20 年中，廉价计算机几乎已经完全取代了 ASCII 终端。然而，即使是图形显示中的“终

www.TopSage.com

端”窗口，使用的驱动程序和配置文件也和实际终端的相同，所以系统管理员仍然需要理解这种古老技术的工作原理。

终端配置包括两项主要的任务：确保给终端关联一个进程以便接受用户的登录，还有就是确保用户一旦登录成功可以得到有关终端的信息。不过，在我们深入介绍这两项任务的细节之前，先让我们看看整个登录过程是怎样的。

27.9.1 登录过程

登录过程牵扯到几个不同的程序，其中最重要的程序就是守护进程 **init**。**init** 的任务之一就是在 **/etc/inittab** 文件打开的每个终端端口上产生一个进程，一般叫做 **getty**。**getty** 设置端口的初始特性（比如速率和奇偶校验），并且显示一个登录提示。参考 2.5.1 节了解有关 **init** 守护进程的知识。

getty 程序的实际名称随 Linux 发行版本的不同而不同，有些发行版本提供了多种实现。Red Hat、Fedora 和 SUSE 使用一种叫做 **mingetty** 的简化版本来受理虚拟终端上的登录。为了管理终端和拨入式调制解调器，它们提供了 Gert Doering 的 **mgetty** 实现。Debian 和 Ubuntu 使用 Wietse Venema 等人编写的一种 **getty**，SUSE 上也提供了这个版本，名字叫做 **agetty**。**mgetty** 基本上已经替代了较老的实现 **uugetty**。

为了区分这么多种类的 **getty**，可以按照复杂程度来考虑它们。**mingetty** 最简单，本质上只是个 **getty** 的占位符。它只能处理 Linux 虚拟控制台上的登录。**agetty** 略微完善一点儿，能处理串口和调制解调器。**mgetty** 是当前的主流。它既能处理传入的传真，也能处理登录，并且能正确地执行锁定和协商，以便让同一个调制解调器既能拨入也能拨出。

在完整的登录过程中先后发生的事情如下：

- **getty** 输出**/etc/issue** 文件中的内容，并显示登录提示；
- 用户在 **getty** 的提示处输入登录名；
- **getty** 用指定的名字作为参数执行 **login** 程序；
- **login** 要求输入口令，并对照**/etc/shadow**[5]验证口令；
- **login** 输出**/etc/motd** 中的每日消息，并且运行一个 shell；
- shell 执行适当的启动脚本[6]；
- shell 显示一个提示符，等待输入。

当用户登出系统的时候，控制权就返回给 **init**，它醒来后在终端端口上派生一个新的 **getty**。

在这一连串的事件中，可配置性最高的地方集中在**/etc/inittab** 文件里（其中定义了系统对 **getty** 的正常配置）以及**/etc/gettydefs** 文件里（有些版本的 **getty** 要在其中查找附加的配置信息）。

27.9.2 /etc/inittab 文件

init 支持多种“运行级”，这些运行级决定了该启用哪些系统资源。总共有 7 种运行级：0～6 级，而 s 级被当作是 1 级的同义词（单用户模式运行）。当离开单用户模式时，除非在**/etc/inittab** 中存在下面提到的 initdefault 字段，否则 **init** 会提示输入一个运行级。然后，**init** 将搜索 **inittab** 文件以找出和指定运行级相匹配的所有行。

经常需要设置运行级，以便您可以有一个只启用控制台的运行级，还有另一个运行级，可以启用

[5] 如果没有使用隐蔽口令，那么可以直接从**/etc/passwd** 文件得到口令。此外，**/etc/passwd** 可能会被管理数据库系统（比如 NIS）替代或者补充。参考第 17 章的内容了解更多信息。

[6] **.profile** 文件用于 **sh** 和 **ksh**；**.bash_profile** 和**.bashrc** 文件用于 **bash**；**.cshrc** 和**.login** 文件用于 **csh/tcsh**。

www.TopSage.com

所有终端。您可以使用任何适合您系统的方式来定义运行级。但是我们建议不要偏离默认设置太远。

inittab 中的配置项采用下面这样的形式：

```
id:run-levels:action:process
```

这里有一个 **inittab** 文件的例子：

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
```

在这个格式下，*id* 是用于标识记录的单字符或者双字符的字符串，它可以为空。对于终端的配置项来说，习惯上是使用终端编号作为其 *id* 的。

run-levels 列举出了当前配置项所从属的运行级。如果没有指定运行级（像第一行那样），那么该配置项对于所有的运行级都有效。*action* 字段表明如何处理 process 字段，表 27.6 列出了一些比较常用的值。

表 27.6 /etc/inittab 文件中 *action* 字段的常用值

值	是否等待	意 义
initdefault	-	设置初始运行级
boot	否	在第一次读取 **inittab** 时运行
bootwait	是	当第一次读取 **inittab** 时运行
ctrlaltdel	否	响应键盘上同时按下<Control-Alt-Delete>键而运行
once	否	启动进程一次
wait	是	启动进程一次
respawn	否	始终保持进程运行
powerfail	否	当 **init** 接收到电源故障信号时运行
powerwait	是	当 **init** 接收到电源故障信号时运行
sysinit	是	在访问控制台之前运行

如果某个 *run-levels* 和当前运行级相匹配，并且 *action* 字段表明了该记录和当前运行级相关，**init** 就会使用 **sh** 执行（或者终止）*process* 字段中所指定的命令。表 27.6 中的“是否等待”这一列表明了 **init** 是否会在继续执行其他命令之前等候命令的完成。

上面给出的 **inittab** 文件的例子中，最后两行在前两个虚拟控制台上派生出 **mingetty** 进程（用 <Alt-F1><Alt-F2>访问）。如果您加上了硬连线的终端或者拨入式调制解调器，那么相应的 **inittab** 配置行看上去和这几行类似。不过，针对这样的设备，必须使用 **mgetty** 或者 **getty**（在 SUSE 上是 **agetty**），因为 **mingetty** 不够完善，不能正确地处理它们。一般而言，`respawn` 是正确的动作，2345 是一组合适的运行级。

命令 **telinit –q** 让 **init** 重新读取 **inittab** 文件。



不同的 **getty** 要求的配置过程也不一样。SUSE、Debian 和 Ubuntu 上都能找到的 **getty/agetty** 版本一般要比 **mgetty** 条理性好些，因为它在命令行（**/etc/inittab** 里）上得到其所有的配置信息。

www.TopSage.com

一般的方式为：

```
/sbin/getty port speed termtype
```

这里的*port*是串口的设备文件（相对于目录**/dev**而言），*speed*是波特率（例如38 400），*termtype*确定了端口的默认终端类型。*termtype*引用了**terminfo**数据库中的某一项。大多数终端仿真器都能模拟DEC VT100，叫做**vt100**。还有许多别的次要选项，其中大多数都和处理拨入式调制解调器有关。参考下一小节了解有关**terminfo**数据库的更多知识。



Red Hat、Fedora和SUSE提供的**mgetty**，在处理调制解调器方面，要比**agetty**稍微好一些，它能把接收和发送传真的能力结合到一起。遗憾的是，它的配置也更分散一些。除了别的命令行标志之外，**mgetty**还可以接受的选项是引用**/etc/gettydefs**文件中的配置行，它指定了串口驱动程序的配置细节。除非您正在设定复杂的调制解调器配置，否则在通常情况下，您不会用到**gettydefs**内的配置行。



在Red Hat系统上，可以用命令**man mgettydefs**来找到**gettydefs**文件的手册页。之所以起这样的名字是为了避免和原来的**gettydefs**的手册页冲突，后者在任何Linux系统里都不再有了。

用于硬连线终端的**mgetty**简单命令行看上去和下面的类似：

```
/sbin/mgetty -rs speed device
```

*speed*是波特率（例如，38 400），*device*是串口的设备文件（采用完整的路径名）。

如果想要在使用**mgetty**时指定端口的默认终端类型，必须在一个单独的文件**/etc/ttytype**中指定，而不能在**mgetty**的命令上这样做。**ttytype**中配置行的格式如下：

```
termtype device
```

*device*是代表端口的设备文件名，*termtype*是恰当的**terminfo**项（参见下一节）的名称。例如：

```
linux    tty1
linux    tty2
vt100    ttyS0
vt100    ttyS1
```

27.9.3 终端支持：termcap和terminfo数据库

Linux通过使用终端功能数据库支持了许多不同的终端种类，这个数据库指出了每种牌子的终端的功能以及在编程方面的特殊之处。历史上一直有两种相互竞争的终端数据库格式：**termcap**和**terminfo**。为了获得最大的兼容性，Linux的各发行版本一般两者都提供。**termcap**数据库保存在文件**/etc/termcap**中，而**terminfo**数据库则保存在**/usr/share/terminfo**目录下。这两种数据库彼此类似，对于每种终端类型，一般用的名称都相同，所以它们之间的区别并不重要。

这两个数据库随操作系统一起发布时，都包含了用于数百种不同终端的条目。在如今这个很少用终端的时代，大多数终端完全用不上。好的做法是，一切都模拟DEC VT100，而经证明别的类型更合适的情况除外。许多终端仿真器也支持“ansi”。“linux”和“xterm”类型的终端分别对Linux控制台和**xterm**（X Windows终端）窗口有用处。

Linux程序检查环境变量TERM来判断您正在使用的是哪一种终端。然后就可以到**termcap**或者**terminfo**里查找它。系统一般在您登录时，根据**getty/agetty**的命令行参数或者**/etc/ttytype**的内容为您设置TERM变量。参考27.12节了解有关登录时配置终端的更多知识。

www.TopSage.com

27.10 特殊字符和终端驱动程序

终端驱动程序支持一些通过在键盘上键入特定键（通常是控制键）就可以访问的特殊功能。使用 **tset** 和 **stty** 命令能够设置功能与键之间的精确绑定关系。表 27.7 列出了其中的一些功能，以及它们默认的绑定键。

表 27.7 终端驱动程序的特殊字符

名 称	默 认 值	功 能
erase	<Control-?>	清除输入的一个字符
werase	<Control-W>	清除输入的一个单词
kill	<Control-U>	清除输入的整行
eof	<Control-D>	发出“文件尾”标志
intr	<Control-C>	中断当前运行的进程
quit	<Control-\>	终止当前进程，并产生一个内存转储文件
stop	<Control-S>	停止屏幕输出
start	<Control-Q>	恢复屏幕输出
susp	<Control-Z>	挂起当前进程
lnext	<Control-V>	按转义前的意义来解释下一个字符

在默认情况下，Linux 的 PC 版内核在按下退格键（这个键可能标为“backspace”或者“delete”，或者只显示一个向左的箭头图案。这取决于键盘）时会产生一个 delete 字符（<Control-?>）。以前，许多 UNIX 系统使用 backspace 字符（<Control-H>）来实现这个功能。遗憾的是，针对这一功能，存在两种不同的标准造成了大量问题。

您可以使用 **stty erase**（参见下面的内容）命令告诉终端驱动程序您的设定实际产生的是哪一个键代码。不过，有些程序（比如文本编辑器以及带有命令编辑功能的 shell）对于 backspace（退格）字符应该是什么有它们自己的想法，它们不一定会关注终端设备驱动程序的设置。对事情有帮助，但又令人比较糊涂的是，有些程序既接受 backspace 字符也接受 delete 字符。您可能还会发现自己通过网络登录进去的远程系统对此的假定与您本地的系统截然不同。

解决这些烦人的小冲突，就任务本身来说挺麻烦。一般说来，没有简单通用的解决方案。每种软件必须单独予以解决。

27.11 stty：设置终端的选项

利用 **stty** 可以直接修改和查询终端驱动程序的各种设置。虽然在这方面有大量的选项，但忽略大多数也没什么坏处。**stty** 使用的驱动程序选项，一般和 **termios** 的手册页中使用的名字一样，但是偶尔也会有差异。

stty 的选项能够以任何顺序和任何组合形式出现在命令行中。在选项之前加短划线可以取消该选项。例如，为了配置终端，使其采用偶校验、不带硬制表（tab），以 9 600 波特率进行运行，可以使用如下命令：

```
$ stty 9600 even -tabs
```

用于基本终端的一种较好的选项组合是：

www.TopSage.com

大家网 TopSage.com

```
$ stty intr ^C kill ^U erase ^? -tabs
```

在这个例子中，**-tabs** 防止终端驱动程序利用终端内置的制表机制，因为很多终端仿真器都不善处理制表，所以这种做法是有用的。其他的选项将中断字符、抹行字符和清除字符的操作分别设置为<Control-C>、<Control-U>和<Control-?>（即退格键）（译者注：如果您在终端的命令行上运行 **reset**，就可以看到上述 3 种最常用的字符和键的映射关系）。

可以使用 **stty** 来检查终端驱动程序的当前模式并且设置它们。不带任何参数的 **stty** 产生的输出如下所示：

```
$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
```

要给出更详细的状态报告，可以使用**-a** 选项：

```
$ stty -a
speed 38400 baud; rows 50; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

输出的格式与不带参数时的类似，只是列出的信息更多。输出的含义应该说是非常直观明显的（如果您最近写过终端驱动程序的话）。

stty 对其标准输入的文件描述符进行操作，所以可以通过使用 shell 的输入重定向符“<”来设置和查询不是当前终端的另一个终端的模式。在大多数系统中，您必须是超级用户才能够修改别人的终端模式。

27.12 tset：自动设置选项

tset 将终端驱动程序初始化为适合给定的终端类型的模式。该类型可以在命令行中指定。如果没有指定，**tset** 将使用 TERM 环境变量中的值。

tset 支持一种可以把 TERM 环境变量中的特定值映射为其他值的语法。如果经常通过调制解调器或者数据交换机登录系统，并且希望为连接的另一端您正在实际使用的终端正确地配置终端驱动程序，而不是一些诸如“dialup”之类普通而没有帮助的配置，那么这一功能就会派上用场。

例如，假如您在家使用 **xterm**，而您正拨入的系统却配置成认为调制解调器的终端类型为“dialup”。那么如果您在自己的**.login** 或者**.profile** 文件中加入命令：

```
tset -m dialup:xterm
```

无论什么时候您拨入系统，终端驱动程序都会为 **xterm** 进行适当地设置。

遗憾的是，**tset** 命令实际上并没有它看上去的那样简单。除了设置您的终端模式，要让 **tset** 调整您的环境变量，还需要用到类似下面的命令行：

www.TopSage.com

```
set noglob
eval `tset -s -Q -m dialup:xterm`
unset noglob
```

这段咒语般的命令使 **tset** 在正常情况下会打印出的消息（**-Q** 标志的作用）不显示，并且让 shell 命令把环境变量 TERM 设置为标准输出（**-s** 标志的作用）。**tset** 打印出的 shell 命令由反引号扩起来，送给 shell，作为内置命令 **eval** 的输入，这就让命令产生的效果仿佛是用户键入它们一样。

set noglob 防止 shell 对 **tset** 的输出中包括的任何元字符（比如“*”和“?”）进行扩展。**sh/ksh** 用户不需要这条命令（也不需要 **unset noglob** 来取消），因为这两种 shell 一般不会扩展反引号中的特殊字符。无论您用的是哪一种 shell，**tset** 命令本身都是一样的；**tset** 查看环境变量 SHELL 来判断要输出什么风格的命令。

27.13 怎样解开僵住的终端

在运行一些程序（比如说 **vi**）时，会对终端驱动程序的状态造成很大的改变。对于用户而言，这种干扰通常是不可见的，因为只要该程序退出或者暂停执行，终端的状态就会完好地恢复成原来的样子。然而，有可能在没有执行恢复处理的时候，某个程序就崩溃或者被杀掉了。当发生这种情况的时候，您的终端可能会出现非常奇怪的行为：它可能会不能正确地处理换行符，不能恰当地回显输入的字符，或者不能正确地执行命令。

另外一个造成终端混乱的常见方式就是偶尔对一个二进制文件运行 **cat** 或者 **more** 命令。大多数二进制文件都包含了特殊字符的混合，足以使一些不够稳定的终端和仿真程序发生混乱。

为了解决这种情况，您可以使用 **reset** 或者 **stty sane**。在大多数系统中，**reset** 实际上只是 **tset** 的一个链接，它能够接受大多数的 **tset** 参数。然而，通常情况下运行它都是不带任何参数的。**reset** 和 **stty sane** 都可以恢复终端驱动程序的正确状态，如果有 **termcap** 和 **terminfo** 中的一个，它们将从 **termcap** 或者 **terminfo** 中发出适当的复位代码。

在很多适合复位操作的情况下，终端已经被置于一种对于您输入的字符不执行任何处理的模式。当按下 Return 或者 Enter 键时，大多数终端都产生一个回车符，而不是换行符，也就是说没有了对输入的处理，该键产生<Control-M>字符，而不是将当前命令送交执行。为了能够直接输入换行符，可以使用<Control-J>或者换行键（如果有的话）来代替 Return 键。

27.14 调制解调器

调制解调器将计算机产生的数字串行信号转换为适合于标准电话线路传输的模拟信号。大量应用中都用到了调制解调器。请参看 12.13 节中的典型例子。

外置调制解调器在模拟信号端有一个 RJ-11 插口，并且在数字信号端有一个某种类型的 RS-232 接口——通常是孔式的 DB-25。它们通常在前面板有一串指示灯，显示了调制解调器当前活动的状态和级别。这些指示灯对于调试极为有用，所以调制解调器一般应该放置在容易看到的地方。

内置调制解调器通常只能在 PC 中看到。它们插入到 ISA、PCI 或者 PCMCIA 插槽中，一旦调制解调器安装完成，它们在计算机机箱背面还会伸出一个 RJ-11 插口。它们比外置调制解调器价格便宜，但配置麻烦，而且通常情况下它们没有指示灯。

如果您正在考虑使用内置调制解调器，那么必须进行检查以确保 Linux 支持它。快速 CPU 通过在

主处理器中执行一些信号处理任务，而有可能简化调制解调器的硬件。遗憾的是，在这种方式下工作的调制解调器（一般叫做 Winmodems）需要非常复杂的驱动程序，而且在 Linux 下，普遍不支持它。

从用户级软件的角度来看，内置调制解调器通常就好像它们是通过一个虚拟串口连接的一般。这种约定有助于把调制解调器的逻辑功能同它的硬件实现隔绝开来。标准的软件包能够驱动调制解调器，而不必知道它的任何特殊之处。

各种调制解调器在健壮性上有所不同，但没有直接的经验是很难判断这一特性的。以前，我们发现某些调制解调器比其他调制解调器对线路噪声有更大的忍受能力。现在，大多数设计都使用了来自几个大型制造商的标准芯片集，所以可能不同的调制解调器之间的差别没有以前那么大了。

高速调制解调器需要复杂的固件，而这些固件偶尔会存有缺陷。如果可能，制造商会在各种型号之间共享固件，所以往往会以产品线的形式成批出现好的或者坏的固件。因为这个原因，我们仍然推荐使用那些著名品牌的产品。

27.14.1 调制、错误纠正和数据压缩协议

很早以前，检查调制解调器到底支持哪些协议是一件非常重要的事情，因为这些标准在不断变化，而调制解调器的制造者们并不总是实现一套完整的协议。现在，给定速率的调制解调器全部都支持相同的协议。它们之间唯一的不同就是固件、电子器件和支持的质量。

协议的波特率就是载波信号被调制的速率。如果有两个以上的信号级别，那么每次传输能够发送不止一个比特的信息，这样以比特每秒计算的速率就会高于波特率。从历史上看，调制解调器的数据速率和信号速率是相同的，从而导致了“baud（波特）”和“bit/s（比特每秒）”这两个名词被人们混用。

现在的调制解调器使用的是“56K”V.90 标准，它实际上并没有提供 56kbit/s 的吞吐率。最好的情况下，它能够允许计算机到 ISP 的速率为 33.6kbit/s，在另一个方向上的速率为 53kbit/s。V.90 标准最近升级到了 V.92，该标准以 48kbit/s 代替 33.6kbit/s 的速度向 ISP 发送数据。V.90 和 V.92 都能够获得与普通音频电话线路中信号传输的理论极限非常接近的速度，并且在近期内是不可能被取代的。

线路噪声会把大量的错误引入到调制解调器的连接中。人们已经开发出各种各样的错误纠正协议，将传输数据打包并且为错误数据提供基于校验和的纠正，把用户或者应用程序从线路错误中解脱出来。以前，您必须知道关于这方面的知识才能正确配置调制解调器，但现在它往往立即就能工作。

数据压缩算法能够用来压缩那些必须在模拟端点之间传输的比特数目。压缩比从最差（当传输的数据已经被压缩过的时候）到最好的情况大约在 4∶1 之间变化。更典型的值为 1.5∶1。大体而言，在启用这些压缩算法中的一种时，普通配置会工作得更好一些。

27.14.2 minicom：拨出程序

虽然传统的 UNIX 拨出程序 **tip** 和 **cu** 都被移植到了 Linux 上（**cu** 通常和 UUCP 一同打包，UUCP 是一种过时的电话通信系统），但是它们在 Linux 系统上没有那么流行。在 Linux 下，更常见的是诸如 **kermit** 和 **minicom** 这样的全套软件包，它们提供了终端模拟功能，而且支持数据传输协议。对于调试以及其他偶尔需要用到的情况，我们之所以推荐采用 **minicom**，大多因为它是最有可能已经预装的软件。Debian 默认没有安装它，如果缺少它，可运行 **apt-get install minicom**。

不管是好是坏，**minicom** 比大多数 Linux 软件都更像 PC 软件。如果您还记得，在 1986 年要以 1200 波特率通过 MS-DOS 终端仿真程序登录系统，而 **minicom** 却能让您轻车熟路，轻松搞定。要配置调制解调器，可以以 root 身份运行 **minicom -s**，进入“serial port setup（串口设置）”菜单。设置调制解调器的设备文件，打开硬件流控，设置 8N1 编码方式（8 位数据，奇数校验，1 位停止位），确保速率正

www.TopSage.com

确。然后返回主菜单，选择“save settings as dfl”，把做的改动记下来。

如果您熟悉大多数调制解调器使用的 Hayes（贺氏）命令语言，那么可以直接输入命令（例如，用“ATDT5551212”拨叫号码 555-1212）。键入<Control-A>和 D 进入拨号菜单，就可以让 **minicom** 为您做这一工作。

27.14.3 双向调制解调器

用一个调制解调器既做拨入又做拨出服务往往是很方便的，如果您想要支持传真的话，那就尤其方便了。这种配置要求 **getty** 以一种特别的方式来处理串口，因为它不能只占着端口不放，不让其他所有进程用端口。通过系统调用 **open** 的选项就能共享端口，这包括串口驱动程序和内核两方面的功能。所有共享端口的程序必须遵循适当的协议。

以前配置双向调制解调器是一件非常麻烦的事情，要做许多特定于系统的调整，而且效果还常常不好。只要您使用 **mgetty** 或者 **uugetty**，主要的 Linux 软件包都能彼此配合运行得很好。一般说来，首推 **mgetty**，它的默认行为就是可以共享端口，所以只要插上用就行了。

27.15 调试串行线

调试串行线并不很困难。一些典型的错误有：

- 忘记告诉 **init** 重新读取它的配置文件；
- 在使用 3 根导线的电缆时，忘记设置软载波；
- 使用错误空接的电缆；
- 颠倒焊接或者钳接 DB-25 连接器；
- 由于接线图错误或者没有接线图，而将设备连接到错误的导线上；
- 不正确地设置终端选项。

对于电缆连线的调试问题而言，断路测试盒（breakout box）是一个不可缺少的工具。它被插入到串行线中，并且显示信号通过电缆时每一个针上的信号。好一些的断路测试盒在两端同时具有针式和孔式连接器，所以它们的位置完全是灵活的。和每一个针（针 2、3、4、5、6、8 和 20）关联的 LED 表明了何时针是活动的。

有些断路测试盒是只读的，并且只允许监视信号；其他断路测试盒可以让您重新进行连线，并且在特定的针上设置电压。例如，如果您怀疑电缆需要空接（交叉的），那么您可以使用断路测试盒来更改实际的电缆连线，将针 2 和针 3 交换，同时也将针 6 和针 20 交换。

不好的断路测试盒可能比不使用任何断路测试盒更差。我们喜欢使用的是 Black Box 生产的 BOB-CAT-B。它是一种很容易使用的断路测试盒，价格大约为 250 美元。请参看第 14.14 节以获取关于 Black Box 的更多信息。

27.16 其他常用的 I/O 端口

对于把低速的外围设备附接到 UNIX 系统中而言，串口曾经是一个不容挑战的标准，但是现在带有 USB（通用串行总线，Universal Serial Bus）的 PC 硬件平台却承担了这项任务。

USB 让传统的串口（和并口）为之逊色。它速度快（高达 480Mbit/s）、结构好，而且使用既简单又便宜的标准电缆。USB 迅速成为大多数外设的标准。

www.TopSage.com

您可以购买一端为 USB 端口另一端为 RS-232C 串口的转换器。这些转换器可以让没有内置串口的新 PC 和老式的串行设备进行通信。我们从 www.keyspan.com 买到了 USB-to-RS-232C 的转换器。2.4.22 版以后的 Linux 内核带有 Keyspan 转换器的驱动程序。

并口在 20 世纪 80 年代很流行，它在概念上和串口类似，但是它们一次传输 8 位数据而不只是 1 位数据。并口过去多见于打印机，但是在 Windows 的世界里，它们以前也用于连接 Zip 和磁带机。这些稀奇古怪的用法都被 USB 取代了，Linux 对于打印机之外的串口设备支持不足。

USB：通用串行总线

USB 是 Compaq、DEC、IBM、Intel、Microsoft、NEC 和 Northern Telecom 等公司共同设计的一个通用的外围设备互连系统。第一个 USB 标准在 1996 年公布。在过去 10 年中，Windows 世界对 USB 的接受程度已经像滚雪球一样迅速增长，所有的新 PC 都带有 USB 端口。大多数新的计算机外围设备也能够找到 USB 的版本。如果想知道关于 USB 更多信息，请参考 www.usb.org 网站上的内容。

USB 是一个很棒的系统。我们认为它有可能在以后的很多年中得以使用。它具备了用户对设备总线所期望的几乎所有属性和功能：

- 极其便宜；
- 最多可以连接 127 个设备；
- 电缆只有 4 根导线，电源、地和 2 根信号线；
- 连接器和连接器的类型是标准化的；
- 连接器非常小，电缆很细而且灵活；
- 不需要切断电源就可以连接或者断开设备；
- 信号的速率最多可以达到 480Mbit/s（USB2.0）；
- 能够使用适配器连接以前遗留下来的串行和并行设备。

USB 甚至能够用作一项 LAN 技术，尽管它实际上并不是为 LAN 而设计的。

Linux 能支持的 USB 设备已经相当广泛。USB 标准定义了几类常用设备的标准接口（比如鼠标、调制解调器以及海量存储设备），因此，通常可以直接使用这些设备。另一方面，诸如摄像机和扫描仪这样的设备需要芯片专用的驱动程序。

在**/proc/bus/usb** 下可以找到可用 USB 设备的信息。带有编号的文件对应于已经连接到系统上的 USB 设备（比如 **001**，代表主机自带的 USB 控制器）。文件**/proc/bus/usb/devices** 提供了有关当前设备数量的详细信息，文件**/proc/bus/usb/drivers** 包含了当前已注册的 USB 设备的名字（无论是否正在使用它们）。这两个文件都可以用 **less** 或者您喜欢的文本编辑器来查看。

虽然需要文件系统**/proc/bus/usb** 支持自动配置功能，而且对调试有帮助，但是内核级的驱动程序一般不会直接用到它。大多数 USB 设备都是通过**/dev** 下传统 UNIX 风格的设备文件来访问的。

最常用 USB 的是数据存储设备，如闪存盘（译者注：即通常所说的 U 盘）。参考 7.9 节了解如何安装 U 盘。参考 www.linux-usb.org 的设备列表，可以了解关于其他类型设备的最新信息。

27.17 习题

E27.1 什么是空调制解调器（null modem）电缆？在连接 DCE 和 DTE 串行设备的时候，该怎么用它？

E27.2 您可以用一条 3 线串口电缆来连接串行调制解调器吗？连接串行打印机呢？为什么（不）可以？

www.TopSage.com

TopSage.com

E27.3 硬件流控是怎样发挥作用的？如果系统不能理解硬件流控，该怎么做？

E27.4 什么是伪终端？什么程序会用到伪终端？

E27.5 写一个 **inittab** 配置项，它可以：

a）运行一个叫做 **server-fallback** 的程序，等待它运行结束，如果电源故障，要立即终止系统；

b）如果一个叫做 **unstable-srv** 的服务器崩溃，则派生一个；

c）运行一个叫做 **clean-temp** 的脚本，在系统每次重新引导的时候删除所有的临时文件。

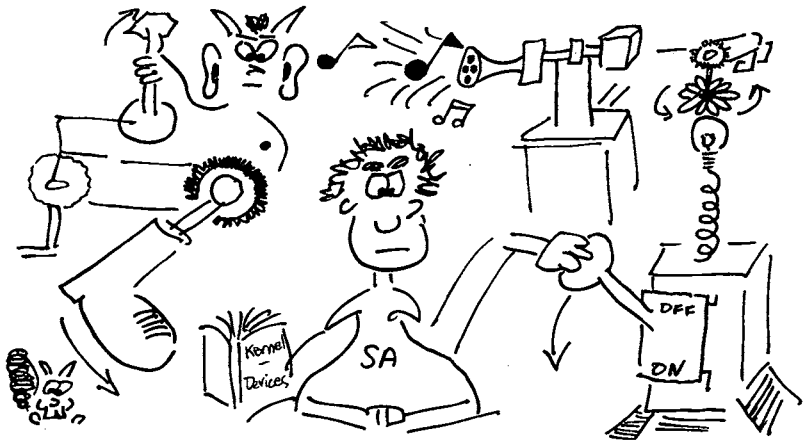
E27.6 您已经把一个新的 USB 设备插到了计算机上。怎样检查该设备已经被识别出来？现在能访问到它吗？

★E27.7 比较 RS-232 和 USB 的串行标准。

★E27.8 您的一个朋友不小心让他自己通宵登录在 Linux 实验室里的系统上，现在当他运行 shell 应用的时候，遇到了奇怪的问题。在遇到某些命令和输入的时候，程序会退出或者挂起，而且以前的输入都不见了，但是有些东西似乎工作正常。是一个不友好的用户导致这样的情况发生吗？阐述您会怎样验证自己的答案。这个问题能够解决吗？谁会做出这样不好的事情？

www.TopSage.com

第 28 章　驱动程序和内核



内核负责把系统的硬件隐藏在一个抽象的高级编程接口之下。它提供了许多在用户和用户级程序看来是理所应当的机制。例如，内核利用下层的硬件功能建立了下面所有这些概念：

- 进程（分时、保护地址空间）；
- 信号和信号量；
- 虚拟内存（内存交换、页面调换、映射）；
- 文件系统（文件、目录、名字空间）；
- 通用的输入/输出（专门硬件、键盘、鼠标）；
- 进程间通信（管道和网络连接）。

内核包含设备驱动程序，它们能管理内核与特定硬件部件的交互，内核的其余部分在很大程度上与设备无关。内核与设备驱动程序的关系类似于用户级进程和内核之间的关系。当一个进程要求内核“读取**/etc/passwd** 的前 64 个字节”时，内核将把这个请求翻译为一条设备驱动程序指令，如“从设备 3 取回第 3348 块”。驱动程序会进一步将这条命令分解成比特位模式的序列，提供给设备的控制寄存器。

内核主要用 C 语言编写，还用到了一点儿汇编语言，帮助它建立和硬件或者特定于芯片的功能之间的接口，这些不能通过通常的编译器指令访问。

Linux 环境的优点之一就是能得到源代码，因此，相对而言容易从头开始编写自己的设备驱动程序和内核模块。在 Linux 的早期时代，系统管理员必须要具有这方面的技术，因为不能为特定的环境量身定制系统的话，想要管理好 Linux 系统会很困难。

如今，系统管理员不必插手棘手的内核代码，就能非常有效地完成管理工作。实际上，有人或许会主张，这类牵扯内核代码的事儿最好留给程序员去做，系统管

www.TopSage.com

理员应该更多地关注用户群体的全面需求。系统管理员可以按本章的介绍，调整内核或者增加现成的模块，但是他们不需要上 C 或者汇编语言编程那么难的课，就能从事系统管理工作。

底线就是，您不应该把现代 Linux 环境的管理和几年以前的管理前沿混为一谈。

28.1 内核的适应性

Linux 系统存在一个可能包含了数以万计不同的计算机硬件的世界中。内核必须适应它所运行的机器上出现的任何硬件。

内核能够以各种方式了解到系统的硬件。对于系统管理员来说，最基本的任务就是要把内核应该找到的（或者可能按有些情况那样，假装没有找到的）硬件的明确信息提供给内核。此外，内核也能在引导时刻，或者系统运行起来以后动态地自己探测一些设备。后一种方法最常用于现代的设备，如 USB 总线上的设备，包括内存棒、调制解调器、数码相机和打印机。还好，Linux 对这类广大的设备都支持得很好。

在 Linux 流行的 PC 平台上，写一份系统硬件的准确清单还是非常困难的（有时候甚至都不可能）。PC 硬件演变的路线并不像自然界的演变，早期的原生生物如今可以进化出从野狗到杀人蜂的各种生物。PC 制造商往往不提供给您有关其销售的系统太多的技术信息，这又加剧了这种差异，所以您不得不经常打开系统，直接用肉眼观察硬件，以找到“主板上的以太网卡用的是什么芯片？”这类问题的答案。

现代 Linux 系统能够运行，依靠的是内核的静态和动态部件的结合。这两者相结合，是由 PC 硬件的局限性造成的。在您系统管理生涯中的某些时候，很有可能会出现您需要通过构建新内核配置的方式来帮忙解决问题的情况。

28.2 添加设备驱动程序

一个设备驱动程序是一个管理着系统与某种特定硬件之间交互作用的程序。驱动程序在设备可理解的硬件命令和内核使用的固定编程接口之间起转换作用。驱动程序层的存在有助于 Linux 合理地保持设备独立性。

设备驱动程序是内核的组成部分，它们不是用户进程。但是，既可以从内核内部，也可以从用户空间访问一个驱动程序。对设备的用户级访问往往要通过位于**/dev** 目录下的特殊设备文件。内核把对这些文件的操作转换为对驱动程序代码的调用。

PC 平台已经成了系统管理员世界里的麻烦之源。有一系列令人炫目的硬件和“标准”，而操作系统对它们支持的程度也各不相同。请看：

- 现在有 30 种以上不同的 SCSI 芯片组，每种芯片组至少又由两倍于上述数字的供应商采用和销售；
- 不同的网络接口有 200 种以上，每种都由若干不同的销售商冠以不同的名称来推销；
- 随时都有更新、更好、更便宜的硬件类型被开发出来。每种都需要一个设备驱动程序以保证在用户所选用的 Linux 上可以使用。

新硬件以惊人的速度发展着，要想使主流的操作系统发布版本和最新的硬件保持同步实际上是不可能的。给内核添加一个设备驱动程序来支持某种新硬件，这实在是很平常的事情[1]。

[1] 对于 PC 硬件，可以使用 **lspci** 命令查看当前接在 PCI 总线上且由内核识别出来的设备。

www.TopSage.com

只有专为 Linux（而且通常是一个特定的 Linux 内核版本）设计的设备驱动程序才能成功地安装到 Linux 系统上。其他操作系统（像 Windows）的设备驱动程序不能用，所以您在购买新硬件时要记得这一点。此外，不同设备用在 Linux 上的时候，其兼容的程度和功能发挥的程度也不一样，所以，对于您正在考虑的硬件，最好留意一下其他用过它的地方使用的结果如何。

供应商对 UNIX 和 Linux 市场越来越重视，他们有时甚至还给自己的产品提供 Linux 驱动程序。如果您很幸运的话，供应商或许会提供驱动程序和安装说明。更可能的情况是，您只能在没有说明的网页上找到您所需要的驱动程序。无论哪一种情况，本节都会告诉您，当您添加一个设备驱动程序的时候到底要如何去做。

28.2.1 设备文件和设备号

许多设备都在**/dev** 目录中有一个对应的文件，而现代操作系统中对此的著名例外就是网络设备。复杂的服务器可以支持数百个设备。

由于是设备文件，在**/dev** 中的每个文件都有与之相关的一个主设备号和一个次设备号。内核用这些设备号把对一个设备文件的引用映射到相应的驱动程序上。

主设备号指定了与文件相关的驱动程序（换句话说，是设备类型）。次设备号常常是指定某种给定设备类型的特定实例。次设备号有时被称为单元号。

用 **ls -l** 可以看到一个设备文件的主设备号和次设备号：

```
$ ls -l /dev/sda
brw-rw---- 1 root   disk    8,   0 Jan 5 2005 /dev/sda
```

这个例子显示了 Linux 系统上的第一个 SCSI 磁盘。它的主设备号为 8，次设备号为 0。

次设备号有时被驱动程序用来选择一个设备的特性。例如，一个磁带驱动器可以在**/dev** 目录中有几个文件，代表记录密度和倒带特性的不同配置。驱动程序可以自由地以它愿意的任何方式解释次设备号。请查阅驱动程序的手册页来确定它所使用的是什么样的约定。

实际上有两种类型的设备文件：块设备文件和字符设备文件。一个块设备每次读取或者写入一块数据（一组字节，通常是 512 的倍数），字符设备每次读取或者写入一个字节。

即使一个设备驱动程序没有控制实际的设备，使用它的抽象实现有时也会比较方便。这种不存在的虚拟设备称为伪设备。例如，通过网络登录的用户会分配到一个 PTY（伪 TTY），从高层软件的角度来看，它各方面都像一个串行端口。这种方法使得在每个人都使用一个 TTY 的时代所编写的程序，仍然可以在窗口和网络时代继续发挥作用。

当一个应用程序对一个设备文件执行一项操作时，内核将自动捕获对该设备文件的引用，并在一张表中检索合适的函数名，然后把控制权交给它。要执行一项不常用的操作（例如弹出一张软盘），且它在文件系统模型中没有直接的类比，则可以使用 **ioctl** 系统调用直接从用户空间向驱动程序传送一条消息。

28.2.2 创建设备文件

设备文件用 **mknod** 命令来创建，语法为：

mknod *filename type major minor*

其中 *filename* 是要创建的设备文件；*type* 为 **c** 代表一个字符设备，为 **b** 代表一个块设备；而 *major* 和 *minor* 分别是主设备号和次设备号。如果要为一个已经在内核中存在的驱动程序创建一个设备文件，

www.TopSage.com

请查看设备驱动程序的手册页，找到合适的主设备号和次设备号。

以前，**/dev** 下的设备文件是由系统管理员手工创建的。大多数系统提供一个名为 **MAKEDEV** 的 shell 脚本（位于**/dev** 目录中），可以帮助完成这项任务。**MAKEDEV** 有时候（但不总是）知道如何为某个特殊部件创建正确的设备文件。以前充其量是个令人厌倦的过程。

到了 2.6 版的 Linux 内核[2]，udev 系统根据设备实际出现（或者消失）的情况，动态地管理设备文件的创建和删除。守护进程 **udevd** 监听内核传来的有关设备状态变化的消息。根据 **/etc/udev/udev.conf** 以及子目录里的配置信息，在找到设备或者断开设备的时候，**udevd** 能够采取各种措施。在默认情况下，**udevd** 创建在**/dev** 里的文件。在检测到新的网络接口的时候，它还会尝试运行网络配置脚本。

28.2.3 sysfs：设备

2.6 版内核引进的另一种特性是 sysfs。这是一种虚拟文件系统，提供了有关可用设备、设备的配置及其状态的组织清晰、内容极其详尽的信息。从内核内部和用户空间都可以访问到这些信息。

您可以浏览 sysfs 文件系统一般安装的**/sys** 目录，在那里找到各种信息，从一个设备正在使用什么 IRQ，到有多少数据库排队等待写入硬盘控制器。sysfs 的指导原则之一就是**/sys** 里的每个文件都只表示下层设备的一个属性。这项约定给原本混乱的数据集赋予了一定的结构性。

能从系统获得的有关设备配置的信息，原本是放在**/proc** 文件系统（本章后面讨论的 profs）的。虽然**/proc** 继续有关于进程和内核在运行时刻的信息，但是特定于设备的信息则应该逐步转移到**/sys** 下。

因为 sysfs 相当新，所有其大量的潜力尚未被发掘出来。最终有可能通过 sysfs 实时地配置设备。长期来看，它甚至会部分或者全部替代**/dev**。只有时间才有发言权。

28.2.4 设备命名约定

设备命名的约定有点儿随意。它们常常沿袭了 DEC PDP-11 上的 UNIX 所用的方式，这在现在这个时代可能听上去显得颇有古风，而且过时了。

串行设备文件的名称通常是 **ttyS** 后跟一个数字，标明该串口所连接的接口。有时 TTY 由不止一个设备文件来代表，多出来的文件通常提供了对其他流控方法或上锁协议的访问。有关串口的更多信息，请参见第 27 章。

磁带设备的名字往往不仅包括对驱动器本身的引用，还包括一个指示，说明该设备在磁带设备关闭之后是否倒带。

IDE 硬盘设备的名字是**/dev/hd**LP，这里的 L 是一个字母，表示设备单元（**a** 表示第一个 IDE 接口上的主[master]设备，**b** 表示那个接口上的从[slave]设备，**c** 表示第二个 IDE 接口的主设备，依此类推），P 是分区号（从 1 开始）。例如，第一个 IDE 硬盘上的第一个分区一般为**/dev/hda1**。SCSI 硬盘的命名方式也类似，但前面是**/dev/sd** 而不是**/dev/hd**。对于这两种类型的设备，您都可以省去分区号来访问整个硬盘（例如，**/dev/hda**）。

SCSI CD-ROM 驱动器由文件**/dev/scd**N 来引用，这里的 N 是一个数字，区分不同的 CD-ROM 驱动器。现代 IDE（ATAPI）CD-ROM 驱动器的引用方式和 IDE 硬盘类似（例如，**/dev/hdc**）。

[2] udev 完全替换了 devfs，后者是最近另一种具有类似功能的尝试。udev 没有实现 devfs 的全部功能，但认为它在体系结构上更单纯。您难道不知道 devfs 的所有那些多出来的方便特性都只是不好的习惯吗？

www.TopSage.com

大家网 TopSage.com

28.3 配置内核的原因和方法

在安装系统的时候，它带有一种通用配置，这是为了在大多数硬件上运行大多数应用而设计的。通用配置包括许多不同的设备驱动程序和选项包，它还为“通用目的”而有选择地调整了参数值。仔细研究这个配置，调整它使之适用于您的确切需要，就能增强系统的性能、安全性乃至可靠性。

在从内存中剔除不想要的驱动程序方面，现代的 Linux 内核要优于它们的前辈，但要一直打开编入（compiled-in）选项。尽管为了提高效率而重新配置内核不如以前那样重要了，但这样做仍会得到良好的效果。

重新配置内核的另一个原因是增加对新设备类型的支持（即添加新的设备驱动程序）。驱动程序代码不能像块橡皮泥那样只是粘到内核上，而是必须整合到内核的数据结构和表中。在某些系统中，这个过程可能需要用户找回内核的配置文件，在其中加入新设备，从头开始重编译内核。在其他系统上，用户可能只需要运行一个专门用于更改这些配置的应用程序就行了。有关添加新驱动程序的信息，请参见 28.6 节。

配置一个内核并不难，只是一旦弄坏了它，再要修好却不容易。

配置 Linux 内核可以采用 4 种基本方法。您可能会有机会在最后试遍所有这 4 种方法。这些方法是：

- 修改可调（动态）的内核配置参数；
- 从头开始构造一个内核（实际上，这意味着从源代码文件开始编译它，可能还要做修改和补充）；
- 随时把新的驱动程序和模块加入现有的内核；
- 通过内核加载程序 LILO 或者 GRUB 在引导时刻提供操作指令。参考 2.3 节了解有关这些系统的更多信息。

这些方法中的每一种所适用的场合都稍有不同。修改可调参数最简单也最常用，而从源代码文件构造内核最困难，也最少需要用到。幸运的是，所有这些方法只要通过一点儿实践就能掌握。

28.4 调整 Linux 内核参数

内核里的许多模块和驱动程序在设计时都知道一个尺寸不能适应全部要求。为了提高灵活性，要有一些特殊的挂钩（hook），能够让系统管理员随时调整诸如内部表的大小或者内核在特殊场合下的行为等这样的参数。通过由文件系统**/proc**（也叫做 `procfs`）中的文件所表示的丰富的“内核到用户”的接口，就能够访问这些挂钩。在许多情况下，大型的用户级应用（特别是诸如数据库这样的“基础设施”级应用）会要求您调整参数来满足它的需要。

/proc/sys 下的特殊文件能够让您查看和设置内核在运行时的选项。这些文件模仿标准的 Linux 文件，但是它们实际上却是进入内核的后门。如果这些文件中的某一个包含您想要修改的值，那么您就可以试着写入它。遗憾的是，不是所有的文件都可写（不管它们表面上的权限是怎样的），而且也没有文档告诉您哪些能写，哪些不能写。如果您有内核源代码，那么可以在子目录 **Documentation/syscnt** 下了解其中一些取值及其含义。

例如，为了改变一个进程能够打开的最大文件数，可以试试：

```
# echo 32768 > /proc/sys/fs/file-max
```

www.TopSage.com

一旦您习惯了这种非正规的接口，您就会发现它相当方便，特别是对于改变配置选项来说更是如此。不过，有一条要注意：重启系统以后原先所做的更改就没了。如果您想要让改动永久生效，就应该使用下面介绍的**/etc/sysctl.conf** 文件。表 28.1 列举了一些有用的选项。

表 28.1 在/proc/sys 下用于某些可调内核参数的文件

目 录[a]	文 件	默 认 值	功 能
C	autoeject	0	在卸载文件系统时会自动弹出光盘吗？凌晨 3 点弹出光盘会把机房里的操作员吓一跳
F	file-max	4096	设置每个进程打开的最大文件数。在包含大量文件的系统上，尝试把这个值增加到 16 384
F	inode-max	16 384	设置每个进程打开的最大索引节点（inode）数。如果您正在写一个应用程序要打开成千上万个文件句柄，那么调整这个参数会有所帮助
K	ctrl-alt-del	0	按下 Ctrl-Alt-Delete 组合键时要重启系统吗？这可能是个个人偏好的问题，对于物理上不安全的服务器控制台来说，这样做或许会提高安全性
K	printk_ratelimit	5	内核消息最小间隔的秒数
K	printk_ratelimit_burst	10	在实际实施 **printk** 速率限制之前接连出现的消息数量
K	shmmax	32M	设置最大共享内存的大小。如果应用程序要求大量的共享内存则要对它进行调整
N	conf/default/rp_filter	0	启用源路由核实。这种防欺骗机制让内核丢弃从“不可能的”路径收到的包
N	icmp_echo_ignore_all	0	在设为 1 时忽略 ICMP ping，设为 1 几乎总是个好主意
N	icmp_echo_ignore_broadcasts	0	在设为 1 时忽略广播 ping，设为 1 几乎总是个好主意
N	ip_forward	0	在设为 1 时允许 IP 转发，如果在把 Linux 主机当路由器用，就要把它设为 1
N	ip_local_port_range	32 768 61 000	指定在建立连接期间分配的本地端口范围。对于要发起许多对外连接的服务器来说，为提高性能可把这个值增至 1 024～65 000
N	tcp_fin_timeout	60	指定等待最后 FIN 包的秒数。在流量大的服务器上设为较低的值（大约 20）会提高性能
N	tcp_syncookies	0	保护不受 SYN 洪泛攻击。如果您怀疑受到拒绝服务（DoS）攻击，那么就打开它

a. F = /proc/sys/fs，N = /proc/sys/net/ipv4，K = /proc/sys/kernel，C = /proc/sys/dev/cdrom。

在大多数系统上，都能找到修改同样参数的一种更稳定的方法，其形式为 **sysctl** 命令。**sysctl** 能够从命令行或者从文件读取一系列 *variable=value* 对来设置各个变量。**/etc/sysctl.conf** 文件默认是在系统引导时读取，它的内容用于设置参数的初始（定制）值。

例如，命令：

```
# sysctl net.ipv4.ip_forward=0
```

关闭了 IP 转发功能。注意，**sysctl** 用的变量名中以圆点（.）替换了目录**/proc/sys** 中的斜线（/）。

28.5 构造 Linux 内核

因为 Linux 演变得如此迅速，所以您最终会面临需要构造 Linux 内核的情况。内核补丁、设备驱动程序和新功能在不断出现。这实在是一把双刃剑。一方面，有“最新和最好”的支持总是会很方便，但另一方面，跟上总是在发展的新东西会相当花时间。但是一旦您成功地构造好内核之后，就会非常

www.TopSage.com

富有成就感，并且会渴望着再度尝试。

如果运行的是一个“稳定”版本的内核，那么要您构造一个自己的内核的可能性就小一些。原来 Linux 采用了这样一种版本升级方案，版本号的第二部分指出了内核是稳定（偶数）的还是开发中（奇数）的。例如，2.6.6 版内核是一个“稳定”内核，而 2.7.4 是一个“开发中”的内核。现在，这个方案没有得到那么严格的遵守，所以您最好查一下网站 kernel.org，看官方对这个问题是怎么说的。如果您没有依靠某个特定的发行版本提供给您内核，那么网站 kernel.org 也是获得 Linux 内核源代码的最好来源。

28.5.1 内核没问题就不要改它

每隔几个月就出现新版的 Linux 内核，而每天都会发布新的驱动程序和补丁，很容易就会对升级打补丁上瘾。毕竟，有什么会比告诉您的用户群，您找到了一种新的内核补丁，要在下午关闭邮件服务器安装它，更让人感到兴奋的呢？有些系统管理员以这种方式证实自己，人人都喜欢当英雄。

在计划升级和修补内核的时候，优秀的系统管理员会在需求和风险之间仔细权衡。诚然，新的内核版本可能是最新和最好的，但是它和目前的版本一样稳定吗？这次更新和打的补丁可以推迟到月底，和另一组补丁一起安装吗？最重要的是，要经得住诱惑，不要让“和别人（这里是指内核高手们）攀比”成为您的用户群的主导兴趣。

好的原则是，只有在您预计取得的成果（可以用可靠性和性能来衡量）超过了实施安装所付出的时间和精力的情况下，才做升级或者打补丁。如果您在对成果进行量化上遇到了麻烦，那么这就预示着可以以后再打这个补丁。

28.5.2 配置内核选项

Linux 内核的配置方式已经走过了很长一段历程，但是同其他有些系统相比，它仍然显得原始。以前，Linux 内核源代码放在**/usr/src/linux** 下，构造一个内核需要有 root 权限。2.4、2.6 版及以后的内核都可以由普通用户构造，因此现在更正确的做法是把内核源代码放在特定于站点的目录里，比如**/usr/local/src/kernel** 或者*~username*/**kernel**。我们在本章里用 *path_to_kernel_src* 代表您为内核源代码所选择的任何目录。无论怎样，您在系统上构造内核之前都要安装内核源代码包，参考 11.1.7 节的内容了解安装软件包的技巧。

配置内核的过程围绕内核源代码根目录（通常是内核源代码目录的根）下的文件**.config** 展开。内核的所有配置信息都是在这个文件中指定的，但是它的格式却有点儿神秘。要解读 *path_to_kernel_src*/**Documentation/Configure.help** 里的指南来了解各种不同的选项代表什么意思。

为了避免人们直接编辑**.config** 文件，Linux 有几种 **make** 的目标体，能让您用不同的界面来配置内核。如果您正在运行 KDE，那么 **make xconfig** 会提供最漂亮的配置界面。类似地，如果您正在运行 GNOME，那么 **make gconfig** 可能是最好选择。这些命令在屏幕上显示出一个图形配置窗口，您可以从中选择要添加到内核的设备（或者编译为可加载模块）。

如果您没有运行 KDE 或者 GNOME，那么可以调用 **make menuconfig** 使用另一种基于 **curses**[3]的字符图形界面。最后，还有一种比较老的风格，即 **make config**，它提示您对每个配置选项进行选择，但事后不能再回头改变您的选择。我们建议，如果系统支持的话，就采用 **make xconfig** 或者 **make gconfig**；否则就用 **make menuconfig**。要避免采用 **make config** 的方式。

[3] **curses** 是一个库，过去用来创建运行在终端窗口内基于文本的 GUI。

www.TopSage.com

如果您要把现有的内核配置用到一个新版内核上（或者说内核源代码树里），那么可以使用命令 **make oldconfig**，读入以前的配置文件，只提新出现的问题。

这些工具只要有上述功能选项可调用，操作起来就很方便。但遗憾的是，如果您想要为多种体系结构或者硬件平台保留几个版本的内核，用它们就比较痛苦了。

上述的几种配置界面都生成类似下面的**.config** 文件：

```
# Automatically generated make config: don't edit
# Code maturity level options

CONFIG_EXPERIMENTAL=y
#
# Processor type and features
#
# CONFIG_M386 is not set
# CONFIG_M486 is not set
# CONFIG_M586 is not set
# CONFIG_M586TSC is not set
CONFIG_M686=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_TSC=y
CONFIG_X86_GOOD_APIC=y
...
```

由此可见，这个文件的内容相当神秘，而且没有说明 CONFIG 标记是什么意思。有时候，您可以推断出它的意思。基本上，每个 CONFIG 行都是指一个特定的内核配置选项。值为 **y** 将把这个选项编译到内核中，而 **m** 会启用这个选项，但却是作为一个可加载模块。

有些东西可以配置成模块，而有些则不能。您必须知道哪样东西是哪种情况，从**.config** 文件是看不出来的。从 CONFIG 标记到有意义的信息之间也没有简单的映射关系。

28.5.3 构建 Linux 内核的二进制文件

在 Linux 内核配置过程中，最重要的部分就是设定一个合适的**.config** 文件，但是您必须再多经过几步，才能把那个文件转变成最后的内核。

下面概括了整个过程：

- 用 **cd** 命令进入内核源代码的目录；
- 运行 **make xconfig**、**make gconfig** 或者 **make menuconfig**；
- 运行 **make dep**（2.6.x 以后的内核不需要）；
- 运行 **make clean**；
- 运行 **make**；
- 运行 **make modules_install**；
- 把 **arch/i386/boot/bzImage** 拷贝为**/boot/vmlinuz**；
- 把 **arch/i386/boot/System.map** 拷贝为**/boot/System.map**；
- 编辑**/etc/lilo.conf**（LILO）或者**/boot/grub/grub.conf**（GRUB），为新内核加一行配置；
- 如果您正在使用的是 LILO，那么运行**/sbin/lilo** 安装重新配置过的引导加载程序。

make clean 这一步并不是必不可少的，但是从一个清理过的构造环境开始一般是个好主意。实际上，许多问题都是由于跳过了这一步而造成的。

www.TopSage.com

28.6 添加 Linux 设备驱动程序

在 Linux 系统上，设备驱动程序一般会以下面 3 种形式之一发布：

- 针对某个特定内核版本的补丁；
- 可加载模块；
- 施加适当修补的安装脚本或者软件包。

在这些形式中，最常见的是针对某个特定内核版本的补丁。如果运气好，可以得到新设备的补丁，那么就应该按照下面的步骤来执行脚本或者安装软件包，这样，内核就可以识别出您的新设备了。

在您有一个针对特定内核版本的补丁时，在大多数情况下，您都可以按照下面的步骤来安装这个补丁[4]：

```
# cd path_to_kernel_src ; patch -p1 < patch_file
```

相对一个有不同次版本号的内核所做的差异文件（diff）可能会失败，但驱动程序应该还可以工作的。

如果不行，可能就必须要手工把新的设备驱动程序集成到内核源代码树里。运气好的话，这么做的原因是您正在编写自己的设备驱动程序，而不是您成为了商业设备供应商没有提供正确的安装脚本的牺牲品。接下来，我们将讲述如何手工将一个假想的网络“snarf”驱动程序添加到内核中。这是一个非常复杂而且单调的过程，与我们已经见到过的其他版本的 UNIX 操作系统相比时尤其如此。

在内核源代码目录树的 **drives** 子目录中，您可以找到与自己用的设备类型对应的子目录。**drivers** 下的目录清单和下面的类似：

```
$ ls -F path_to_kernel_src/drivers
acorn/      char/      i2c/        Makefile   net/      s390/      telephony/
acpi/       dio/       ide/        md/        nubus/    sbus/      usb/
atm/        fc4/       ieee1394/   media/     parport/  scsi/      video/
block/      gsc/       input/      message/   pci/      sgi/       zorro/
bluetooth/  hil/       isdn/       misc/      pcmcia/   sound/
cdrom/      hotplug/   macintosh/  mtd/       pnp/      tc/
```

最常添加驱动程序的目录有 **block**、**char**、**net**、**scsi**、**sound** 和 **usb**。这些目录分别包含了块设备（例如 IDE 磁盘驱动器）、字符设备（例如串口）、网络设备、SCSI 卡、声卡和 USB 设备的驱动程序。其他一些目录包含了总线本身的驱动程序（例如，**pci**、**nubus** 和 **zorro**），用户不太可能会向这些目录添加驱动程序。有些目录包含了针对平台的驱动程序，例如 **macintosh**、**s390** 和 **acorn**。有些目录包含了一些特殊设备，例如 **atm**、**isdn** 和 **telephony**。

由于我们举例的设备是与网络相关的设备，所以我们将把驱动程序添加到 **drivers/net** 目录中。我们需要修改下面这些文件：

- **drivers/net/Makefile**，这样我们的驱动程序会被编译；
- **drivers/net/Config.in**，这样我们的设备能出现在配置选项中。

在把这个驱动程序的**.c** 和**.h** 文件都放到 **drivers/net/snarf** 下之后，我们要在 **drivers/net/Makefile** 里加上这个驱动程序。我们（在靠近文件末尾的地方）加上一行：

```
obj-$(CONFIG_SNARF_DEV) += snarf/
```

这样的配置把驱动程序 snarf（放在 **snarf/**目录下）加入到了构造内核的过程里。

在把设备加到 **Makefile** 文件以后，我们必须保证在配置内核的时候配上这个设备。所有的网络设

[4] 当然，在您修改内核代码树之前，必须安装好内核源代码包。

www.TopSage.com

备都必须列在文件 **drivers/net/Kconfig** 里。要加上这个设备，让它可以编译成为一个模块或者内核的一部分（与我们在 **Makefile** 文件里的要求保持一直），就要加上下面这两行：

```
config SNARF_DEV
tristate 'Snarf device support'
```

config 后的第一个标记是配置宏，它必须和 **Makefil** 里 CONFIG_后面的标记一致。关键字 tristate 的含义为，我们可以把这个设备编译成一个模块。如果设备不能编译成一个模块，我们就要用关键字 bool 代替 tristate。下一个标记是在配置窗口上显示的字符串。它可以是任意的文本，但是应该说明正在配置的是什么设备。

知道设备

在把一个新设备的驱动程序加入到内核里之后，该如何告诉内核它需要使用这个新的驱动程序呢？在 2.6 版内核之前，这是一个需要掌握编程知识的枯燥任务。作为对设备的驱动程序模型体系结构调整的一部分，现在有一种标准的方法，能把驱动程序同内核联系起来。

解释其中的细节超出了本书的内容范围，但结果是 2.6 版（及以后）内核的设备驱动程序都用宏 MODULE_DEVICE_TABLE 自行登记。这个宏建立正确的幕后联系，像 **modprobe**（下面讨论）这样的其他工具就能启用内核中的新设备。

28.7 可加载内核模块

可加载内核模块（Loadable kernel module，LKM）能够在内核正运行的时候，把驱动程序——或者其他内核服务——链入内核，或者从内核删除。这种方式能够使设备驱动程序的安装更加容易，因为不必改变内核的二进制文件。它还可以使内核变得更小，因为驱动程序只有在需要的时候才被加载。

通过内核明确提供的一个或者多个“挂钩（hook）”，让更多的设备驱动程序可以挂接到内核上，这就实现了可加载模块。用户级命令 **insmod** 能与内核进行通信，告诉内核把新模块加载到内存里。而 **insmod** 命令则卸载驱动程序。

尽管可加载的驱动程序非常方便，但它并不是 100%安全。用户加载或者卸载模块的任何时候，都有可能造成内核恐慌（panic）[5]。如果您不希望让机器崩溃，那么我们建议不要加载或者卸载一个未经测试的模块。

在 Linux 下，几乎任何东西都可以构造成一个可加载内核模块。根文件系统、根文件系统所在的设备以及 PS/2 鼠标驱动程序则是例外。

按照惯例，可加载模块放在**/lib/modules/***version* 目录中，这里的 *version* 是执行 **uname –r** 所返回的 Linux 内核版本号。用户可以用 **lsmod** 命令来检查当前加载的模块：

```
# lsmod
Module          Size  Used by
ppp            21452  0
slhc            4236  0 [ppp]
ds              6344  1
i82365         26648  1
pcmcia_core    37024  0 [ds   i82365]
```

[5] 这种风险非常小，小到在您的环境中可以忽略不计。

www.TopSage.com

这台机器加载了 PCMCIA 控制器模块、PPP 驱动程序以及 PPP 报头压缩模块。

Linux 的 LKM 可以手工加载到内核中。例如，我们可以用下面的命令手工插入我们举例的 snarf 模块：

```
# insmod /path/to/snarf.ko
```

我们也可以给可加载内核模块传递参数，例如：

```
# insmod /path/to/snarf.ko io=0xXXX irq=X
```

一旦已经把可加载内核模块手工添加到内核中，只有明确请求删除它或当系统重新启动的时候，才能把它删掉。我们可以用 **rmmod snarf** 命令来删除 snarf 模块。

您可以在任何时候使用 **rmmod** 命令，但它只在模块的当前引用数（在 **lsmod** 命令的输出中由 `Used by` 这一列给出）为 0 时才会起作用。

Linux 的 LKM 也可以用 **modprobe** 命令半自动地加载，这个命令封装了 **insmod** 命令，它可以懂得依赖性、选项以及安装和卸载步骤。**modprobe** 用文件**/etc/modprobe.conf** 来配置如何处理各个模块。

您可以运行 **modprobe-c** 命令来动态地创建一个与当前所有已安装模块相对应的**/etc/modules.conf** 文件。这个命令将产生一个很长的文件，和下面的类似：

```
#This file was generated by: modprobe -c
path[pcmcia]=/lib/modules/preferred
path[pcmcia]=/lib/modules/default
path[pcmcia]=/lib/modules/2.6.6
path[misc]=/lib/modules/2.6.6
...
# Aliases
alias block-major-1 rd
alias block-major-2 floppy
...
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
...
alias dos msdos
alias plip0 plip
alias ppp0 ppp
options ne io=x0340 irq=9
```

这里的 path 语句说明了在哪里可以找到某个特定模块。如果需要将模块放在一个非标准的位置，用户可以修改或者添加这种类型的项。

alias 语句提供了块主设备号、字符主设备号、文件系统、网络设备、网络协议和与其对应的模块名之间的一种映射关系。

`options` 语句不是动态生成的。它们指定了加载模块时应该传递给模块的选项。例如，我们可以用下面的命令告诉 snarf 模块正确的 I/O 地址和中断向量[6]：

```
options snarf io=0xXXX irq=X
```

modprobe 还能理解 `install` 和 `remove` 语句。当从运行的内核中加载或者卸载某个特定模块时，这些语句可以让这些命令得以执行。

[6] 如果您用的是 PC 硬件，那么创建一个 IRQ（设备中断请求向量，device interrupt request vector）和 I/O 端口不重叠的配置则是一种挑战。您可以分别研究一下**/proc/interrupts** 和**/proc/ioports** 里的内容，了解系统当前的分配情况。

www.TopSage.com

28.8 热插拔

可加载的内核模块满足了内核动态加载和卸载设备驱动程序的需要，而 Linux 的热插拔特性能够将设备是否可用的信息导出到用户空间内。这个功能可以让用户进程对某些事件做出反映，如系统连上了一个 USB 的数码相机或者 PDA。例如，您可能想自动把图像拷贝到本地硬盘或者对您的 PDA 日程表做热同步。热插拔项目的思路为，除了插上设备之外，无需用户额外的输入。

从 2.6 版内核开始，在已经设计使用 sysfs 的总线和驱动程序上都能用热插拔。热插拔、sysfs 和前面讨论过的设备驱动程序的登记有密切的关系。

在当前的热插拔实现中，内核只要检测到插上或者拔掉了一个设备，就会执行由**/proc/sys/kernel/hotplug** 参数指定的用户进程（通常是**/sbin/hotplug**）。**/sbin/hotplug** 是一个 shell 的脚本，它调用**/etc/hotplug/**目录下一个针对设备类型的代理（agent）程序来处理发生的事件。例如，如果加入了一个网络接口，那么就执行**/etc/hotplug/net.agent** 这个脚本，把接口启动起来。您可以增加或者编辑**/etc/hotplug** 目录下的脚本，定制您系统上的热插拔操作。

出于安全和其他原因，您可能不想让热插拔系统对某个事件采取行动。在这类情况下，您可以把设备加入到**/etc/hotplug/blacklist** 文件里，防止和它们有关的事件触发处理行动。

相反，您可以通过创建“handmap”文件，如**/etc/hotplug/*type*.handmap** 强行采取热插拔操作（如果您在创建新名字，要保证 *type* 不会和现有的名字冲突）。

28.9 设置内核引导参数

当您有了一个正常工作运行的内核之后，可能需要在引导内核的时候给它传些特殊的配置选项，比如它应该使用的根设备，或者一条探测多个以太网卡的指令。引导加载程序（LILO 或者 GRUB）负责把这些选项传给内核。

要是选项应该在系统每次引导的时候使用，可以在**/etc/lilo.conf** 或者**/boot/grub/grub.conf** 中加上静态的配置，至于到底是哪个则取决于您选用的引导加载程序是什么。参考 2.3 节了解更多知识。

如果不可能编辑引导加载程序的配置文件（或者您弄坏了什么，机器启动不起来了），可以手工把这些选项传给内核。例如，在 LILO 的引导提示下，您可以键入：

```
LILO: linux root=/dev/hda1 ether=0,0,eth0 ether=0,0,eth1
```

来告诉 LILO 加载由“linux”标记所指定的内核，使用/dev/hda1 为根设备，并检测两个以太网卡。

采用 GRUB 的相似例子就像下面：

```
grub> kernel /vmlinuz root=/dev/hda1 ether=0,0,eth0 ether=0,0,eth1
grub> boot
```

在另一种常见的情形下，使用引导时刻选项也有帮助，就是当检测到 SAN（storage area network，存储区域网）上的 LUN（logical unit number，逻辑单元号）时。在默认情况下，Linux 内核只探测 LUN 0，如果您的环境里用不同的 LUN 表示逻辑存储区域，这就不够用了（联系您的 SAN 管理员或者厂家看是不是这种情况）。在这种情况下，需要告诉内核，在系统引导过程中要检测到多少个 LUN。

例如，如果您想要一个 2.4.x 内核探测 8 个 LUN，那么引导行如下：

```
grub> kernel /vmlinuz root=/dev/hda1 max_scsi_luns=8
```

www.TopSage.com

```
grub> boot
```

在 2.6.x 及以后的内核，参数名变了：

```
grub> kernel /vmlinuz root=/dev/hda1 max_luns=8
grub> boot
```

28.10 推荐读物

BECK, MICHAEL, ET AL. *Linux Kernel Internals, (3rd Edition).* Sebastopol, CA: O'Reilly Media, 2006.

28.11 习题

E28.1 介绍一下内核是干什么用的。说明使用模块的内核和不使用模块的内核之间有什么区别。

★E28.2 用**/proc** 方法和 **sysctl** 方法考察表 28.1 中几个参数的值。用一种方法改变这些值里的两个，然后用另一种方法再读一下。验证系统已经响应您所作的调整，它的行为实际发生了变化。给出您做的试验报告（需要 root 权限）。

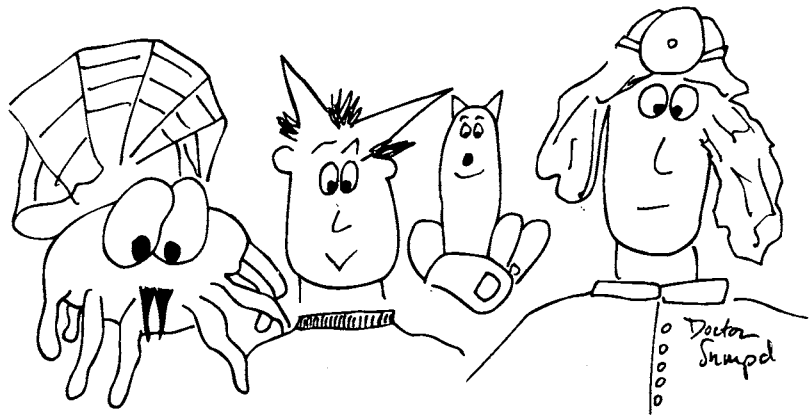
★E28.3 在本地的跳蚤市场上，您可以很划算地买到一块笔记本的网卡，用它可以通过并口连接以太网。让 Linux 认出这个新卡需要执行哪些步骤？应该直接把支持编译到内核中，还是把它当作模块来添加呢？为什么？（多问一句，如果您每小时的咨询费是 80 美元，那么估计一下，为了让这块便宜的以太网卡工作，要花多大价值的工作量。）

★E28.4 刚刚新出了一个 Linux 内核，而您想要把学生实验室里的所有机器都升级一遍（大约有 50 台机器，并不是所有的机器都一样）。您应该考虑哪些问题？应该遵照什么样的步骤？可能出现什么样的问题，如何解决它们？

★★E28.5 在实验室里，用 **xconfig** 或者 **menuconfig** 配置一个内核，构造一个内核的二进制文件。安装并运行新系统。观察新老内核的 **dmesg** 输出，突出其中的不同之处（需要 root 权限）。

www.TopSage.com

第 29 章　守护进程



守护进程是执行一种特定功能或者系统任务的后台进程。同 UNIX 和 Linux 模块化的基本思想保持一致的是，守护进程是程序而不是内核的组成部分。许多守护进程在系统引导时启动，并且只要系统不停机它们就会连续运行。其他守护进程在需要的时候才启动，并且只在它们能派上用场的时候运行。

首次将“daemon（守护进程）”一词用作计算机术语的是一位英国绅士 Mick Bailey，在 20 世纪 60 年代初期，Mick Bailey 在美国麻省理工学院（MIT）从事 CTSS 编程工作[1]。Mick 援引了牛津英语词典来支持这个单词的拼写和含义。单词“daemon”和“demon”都源自于相同的词根，但“daemon”是更古老的拼法，并且它的意思稍有不同。daemon（守护神）是影响人们性格或者个性的服侍神。守护神不是善良或邪恶的奴婢，它们是具有独立思想和意愿的圣灵。daemon（守护进程）的说法从 CTSS 沿用到 Multics，又用到 UNIX，再到 Linux，它们在 UNIX 和 Linux 上非常流行，以至于需要一个超级守护进程（**inetd** 或者 **xinetd**）来管理它们。

本章简要概述了大多数常见的 Linux 守护进程。这里列举的所有守护进程并不是 Linux 的所有发行版本都能提供的，而有些 Linux 的发行版本所能提供的每个守护进程也并没有都在这里列出。除了更清楚地知道 Linux 怎样工作以外，当您的一个用户问起“**xkogd** 是干什么的？”时，您对所有各种守护进程的知识就能让您显得实在是高明。

在编写出 **inetd** 之前，所有守护进程都在系统引导的时刻启动，并且连续运行（或者，更准确地说，它们阻塞着等待任务）。随着时间的推移，系统中加入了越来越多的守护进程。守护进程的总数变得太多，开始引起了性能问题。为了适应这种情况，伯克利的行家们开发出了 **inetd**，这个守护进程负责只有在需要时启动别的守

[1] 这段历史来自麻省理工学院的 Jerry Saltzer，由 Dennis Ritchie 转述。

www.TopSage.com

护进程。**inetd** 成功地普及了超级守护进程的模型，成为把服务器上正在运行的进程数量降至最低的常用方式。现在 UNIX 和 Linux 的大多数版本都把 **inetd** 和一直保持运行的守护进程组合起来使用。

之所以有那么多系统管理员应该密切熟悉的守护进程，不是因为它们在管理上需要许多关照，就是因为它们在系统的日常运行中扮演着重要的角色。我们在这里用一两句话来说明的一些守护进程，在本书的其他地方我们可能会用一整章来介绍。我们在适当的地方给出了交叉索引。

我们在本章一开始先介绍两个非常重要的系统守护进程（**init** 和 **cron**），然后转而讨论 **inetd** 和 **xinetd**。最后，我们简要描述系统管理员在我们所举的 4 种发行版本上有可能会遇到的大多数守护进程。

29.1 init：基本守护进程

init 是系统引导后第一个要运行的进程，而且从许多方面来说，它都是最重要的守护进程。它的 PID 总是 1，它是所有用户进程的祖先，并且是除几个之外其他所有系统进程的祖先。

在启动时，**init** 要么将系统置于单用户模式，要么开始执行将系统带入多用户模式所需的脚本。在引导系统进入单用户模式时，**init** 在终止——这可以通过键入 **exit** 或者按<Control-D>来实现——单用户 shell 之后就运行启动文件。

在多用户模式下，**init** 负责保证在每个开启登录功能的设备上能有进程来处理登录。在串口上的登录一般是由 **getty** 的某个变体（例如，**agetty**、**mgetty** 或者 **mingetty**，参见 27.9.2 节了解详情）来处理的。在许多系统上，**init** 也能监管图形化的登录过程，让用户直接登录到 X Windows 里。

init 除了它的终端管理职责之外，还有一个吓人的驱邪任务，即驱除未死的僵进程，否则这些僵进程就会积累在系统上。**init** 在这个过程中的任务已在 4.1.6 节介绍过。

init 定义了几个“运行级”，这些“运行级”确定应该启用什么样的系统资源集合。运行级通常有 7 个：0～6。模式“s”被当作是运行级 1（单用户模式）的同义词。每个运行级的特性在**/etc/inittab** 文件中定义。参考 27.9.2 节了解有关 **inittab** 文件的更多资料。

通常将 **init** 的初始运行级作为一个参数从引导加载程序传递给它的。如果指定为“s”，则 **init** 进入单用户模式。否则，它就扫描**/etc/inittab** 文件以寻求应用于所需运行级的配置项，并且执行与配置项相对应的命令。

系统启动之后，使用 **telinit** 命令可以改变 **init** 的运行级。例如，**telinit 4** 强制 **init** 进入运行级 4（在我们所举例的系统上不用这个运行级）。**telinit** 最有用的参数是 **q**，这个参数会让 **init** 重读**/etc/inittab** 文件。

Linux 的发行版本在 **init** 提供的基本运行级机制之上又实现了一种附加的抽象层。这多出的一层允许各个软件包能安装它们自己的启动脚本，而不必修改系统的通用 **inittab** 文件。给 **init** 一个新的运行级，就会以 **start** 或 **stop** 为参数执行相应的脚本。

2.5.1 节更完整地讨论了 **init** 和启动脚本。

29.2 cron 和 atd：调度命令

守护进程 **cron**（在 Red Hat 上叫做 **crond**）负责在预定时间运行命令。它从用户和管理员那里接受时间安排文件（“crontab”文件）。

cron 经常用于管理目的，包括管理记账和日志文件，以及文件系统的日常清理工作。实际上，**cron** 对于系统管理员来说是如此重要，以至于我们用整整一章——第 8 章——来介绍它。

www.TopSage.com

守护进程 **atd** 用 **at** 命令运行。大多数的 Linux 版本也包含 **anacron** 这个调度程序，它以一定时间间隔而不是在特定的时间来执行任务。在那些不总是打开的系统上（比如笔记本电脑），**anacron** 特别有用处。

29.3 inetd 和 xinetd：管理守护进程

xinetd 和 **inetd** 是管理其他守护进程的守护进程。当有工作需要它们的客户守护进程去做时，它们就启动这些客户进程，一旦客户进程的任务完成了，它们就让客户进程正常地终止。

传统版本的 **inetd** 是从 UNIX 系统来的，但是大多数 Linux 发行版本已经换到了 Panos Tsirigotis 的 **xinetd**，它是一种更有吸引力的选择方案，其中融入了类似于以前通过使用 **tcpd**——"TCP wrappers" 软件包——才能获得的那些安全特性。**xinetd** 也能针对拒绝服务攻击提供更好的保护，它有更好的日志管理功能以及更为灵活的配置语言。

遗憾的是，**inetd** 的配置文件不能向前兼容 **xinetd**。我们首先讨论较常用的 **xinetd**，然后在另一小节里看看 **inetd**。

在我们举例的 Linux 系统中，只有 Debian 和 Ubuntu 使用标准的 **inetd**，RHEL、Fedora 和 SUSE 都默认采用 **xinetd**。您可以让任何系统转而使用非默认的守护进程管理程序，但确实没有什么非这样做不可的理由。

xinetd 和 **inetd** 都只能对通过网络提供服务的守护进程起作用。为了掌握有人尝试访问客户守护进程的时间，**xinetd** 和 **inetd** 把自己绑定到正常情况下由不活动的守护进程所管理的网络端口上。当出现一条连接请求时，**xinetd/inetd** 就启动适当的守护进程，把它的标准 I/O 通道连接到网络端口上。如果守护进程要能与这种方式兼容，它们就必须按照这种思路来编写。

有些守护进程（例如那些与 NIS 和 NFS 有关的守护进程）还要再依赖更深一层的 RPC（Remote Procedure Call，远程过程调用）系统。RPC 最初是由 Sun 作为一种促进异构网络环境下信息共享的方式而设计和实现的。使用 RPC 的守护进程的端口分配由 **portmap** 守护进程进行管理，本章的后面将讨论这个守护进程。

有些守护进程既能够以传统的方法（在传统方法中，守护进程一旦启动就连续运行，直到系统关闭）使用，也能够和 **xinetd/inetd** 一起使用。在本章中讨论的守护进程，如果是与 **xinetd/inetd** 相容的，那么就标为①。

因为 **xinetd/inetd** 负责管理许多常用的基于网络的服务，所以它在保护系统安全上扮演了重要角色。确认您需要的服务并给予认可是很重要的。在一个新系统上，几乎肯定要修改默认配置，禁用在您的环境中不必要的或者不想要的服务。

29.3.1 配置 xinetd

xinetd 的主要配置文件传统上是**/etc/xinetd.conf**，而各个 Linux 发行版本也都提供一个配置目录**/etc/xinetd.d**。单个软件包可以把 **xinetd.conf** 格式的配置文件放在 **xinetd.d** 这个目录下，不必担心会覆盖掉其他软件包的配置。

下面的例子显示了在 Red Hat Enterprise 系统上一个 FTP 服务的默认参数设定和配置：

```
defaults
{
    instances       = 60
    log_type        = SYSLOG authpriv
    log_on_success  = HOST PID
```

www.TopSage.com

```
    log_on_failure  = HOST
    cps             = 25 30
}

service ftp
{
    # Unlimited instances because wu.ftpd does its own load management
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = root
    server          = /usr/sbin/wu.ftpd
    server_args     = -a
    instances       = UNLIMITED
    only_from       = 128.138.0.0/16
    log_on_success  += DURATION
}

includedir /etc/xinetd.d
...
```

表 29.1 给出了一个小小的参数表。

表 29.1　　xinetd 配置参数（不全）

参　数	取　值	含　义
bind	IP 地址/主机	这个服务所在的接口
cps	数字	限制每秒连接总数
disable	yes/no	禁用服务，比注释掉它更容易
include	路径	读取列出的路径，作为对配置文件的补充
includedir	路径	读取指定目录下的所有问题
instances	数字/ UNLIMITED	并发实例的最大数量
log_on_failure	特殊[a]	故障或者拒绝访问的日志信息[b]
log_on_success	特殊[a]	成功连接的日志信息[b]
log_type	特殊[a]	配置日志文件或者 syslog 参数
max_load	数字	如果负载>阈值，则禁用服务
nice	数字	派生出的服务器进程的谦让值
no_access	匹配列表	拒绝向指定的 IP 地址提供服务
only_from	匹配列表	只接受从指定地址来的请求
per_source	数字	限制每个远程对端的实例数量
protocol	tcp/udp	服务协议
server	路径	服务器二进制文件的路径
server_args	字符串	服务器的命令行参数[c]
socket_type	stream/dgram	TCP 服务用 stream，UDP 服务用 dgram
user	用户名	服务器应该以这个用户（UID）的身份运行
wait	yes/no	**xinetd** 应该在守护进程退出之前不插手服务吗

a. 从一张规定好的表中选出一个或者多个值，太长无法在这张表里再写一遍。

b. 注意，与这些参数连用的 USERID 指令会让 xinetd 对连接执行 IDENT 查询，这经常会造成明显的延迟。

c. xinetd 和 inetd 不同，它不要求第一个参数是服务器命令。

有些 **xinetd** 参数可以接受+=或者-=这种形式的赋值方式（在 FTP 服务器的 `log_on_success` 值中可以看到），对默认值进行修改，而不是彻底替换掉它们。每项服务实际只要求几个参数。

www.TopSage.com

参数 only_from 和 no_access 里的地址匹配列表可以采用几种格式来指定。最有用的是 CIDR 格式，即带有明确子网掩码的 IP 地址（如本例所示），以及主机或者域名，比如 boulder.colorado.edu 和.colorado.edu——注意前面有个点（.）。多个匹配说明则以空格分隔（所有的 **xinetd** 列表都是这样）。

xinetd 既可以把日志直接记入一个文件，也可以把日志项提交给 syslog。因为在一台繁忙的服务器上，日志的信息量有可能变得相当大，所以出于性能的原因，使用“直接到文件”的日志记录方式可能会有意义。要记住，把日志记录到一个文件要比通过 syslog 记录到一台远程服务器更不安全，因为获得对本地文件系统访问权力的黑客能够对日志文件动手脚。

xinetd 可以提供一些有意思的服务，比如转发针对内部（外界不可见）主机的请求。值得研究一下 **xinetd** 的手册页，对它的能力有个概念。

29.3.2 配置 inetd



Debian 和 Ubuntu 是仅有的还在使用传统 **inetd** 的主要 Linux 发行版本。这个版本的 **inetd** 查阅文件**/etc/inetd.conf** 来确定它应该监听哪些网络端口。这个配置文件包括大量与 **xinetd.conf** 相同的信息，但它采用了表格（而不是属性/值列表）的格式。下面是一个 Debian 系统上的例子（经过删节）：

```
# Example /etc/inetd.conf - from a Debian system

#:INTERNAL: Internal services
#echo      stream  tcp     nowait    root    internal
#echo      dgram   udp     wait      root    internal
...
#time      stream  tcp     nowait    root    internal
#time      dgram   udp     wait      root    internal

#:STANDARD: These are standard services.
#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.
imap2      stream  tcp     nowait    root    /usr/sbin/tcpd /usr/sbin/imapd
imaps      stream  tcp     nowait    root    /usr/sbin/tcpd /usr/sbin/imapd

#:INFO: Info services
ident      stream  tcp     wait      identd  /usr/sbin/identd identd
...
#:OTHER: Other services
swat       stream  tcp     nowait.400 root   /usr/sbin/swat swat
# finger   stream  tcp     nowait    nobody  /usr/sbin/tcpd in.fingerd -w
391002/1-2 stream  rpc/tcp wait      root    /usr/sbin/famd fam
...
```

第 1 列是服务名称。**inetd** 通过查阅**/etc/services** 文件（对于 TCP 和 UDP 服务）或者**/etc/rpc** 文件和 **protmap** 守护进程（用于 RPC 服务），把服务名映射成端口号。RPC 服务由 *name/num* 形式的名字以及在第 3 列中的 rpc 标志来标出。上面的例子中，最后一行是一个 RPC 服务。

常由 **inetd** 管理的唯一一个别的 RPC 服务是 **mountd**，它是 NFS 安装守护进程。Linux 的发行版本似乎是以老式的方法（在系统引导的时候启动它）来运行这个守护进程的，所以在您的 **inetd.conf** 文件里，可能根本就没有 RPC 服务。

在带有多个网络接口的主机上，在服务名之前，可以加上若干以逗号隔开的 IP 地址或者主机名代号，指出 **inetd** 应该监听哪些接口上的服务请求。例如，下面这行：

```
inura:time    stream   tcp   nowait   root    internal
```

www.TopSage.com

只在 DNS、NIS（或者/etc/hosts 文件）中的名字为 inura 的接口上才提供时间（time）服务。

第 2 列指定了服务使用的套接口类型，它不是 stream 就是 dgram 类型。stream 用于 TCP（面向连接的），dgram 用于 UDP。

第 3 列标明了服务所使用的通信协议。在 **protocols** 文件（通常在/etc 目录下）中列出了可以使用的类型。协议几乎总是 tcp 或者 udp。RPC 服务在协议类型前冠以 rpc/，如上面例子里的 rpc/tcp。

如果被描述的服务能够一次处理多个请求（而不是处理一个请求就退出），那么应该把第 4 列设置为 wait。这个选项让派生出的守护进程只要在运行就接管端口，**inetd** 等着这个守护进程退出，然后继续监视端口。wait 的反面是 nowait，它让 **inetd** 连续监视端口，在它每次接收到一个请求的时候就派生出一个守护进程的新副本。是选择 wait 还是 nowait 必须和守护进程的实际行为相对应，不应该随便设置。在配置一个新守护进程的时候，要查看 **inetd.conf** 里配置行的样例，或者参考该守护进程的手册页。

在 **swat** 的配置行里出现的 nowait.400 形式，表示 **inetd** 应该每分钟派生这个服务器守护进程的至多 400 个实例。默认值更保守，是每分钟 40 个实例。鉴于这项服务的性质（Samba 的 Web 管理工具），我们不清楚为什么要提高这个阈值。

第 5 列给出了运行守护进程的用户名。如果您不信任某个特定程序或者您知道它有安全问题，那么就可以以其他人的身份而不是以 root 的身份来运行它，从而减少安全隐患。当然这种技巧仅适用于不需要 root 权力的守护进程。在上面的例子中，**in.fingerd** 以“nobody”用户的身份运行。

剩下的几列给出了守护进程的完整路径及其命令行参数。关键字 internal 指出服务的实现由 **inetd** 自己提供。

在这个例子里的大多数服务都通过 **tcpd** 的方式来运行它们的守护进程，而不是直接运行它们。**tcpd** 要对连接企图做日志记录，并且根据连接企图的来源实现访问控制。一般而言，所有的服务都应该用 **tcpd** 予以保护。这个配置的例子有一个潜在的安全问题，因为文件共享配置工具程序 **swat** 没有受到保护[2]。

在 Debian 带的默认 **inetd.conf** 文件里，**rlogin**、**telnet**、**finger** 和 **rexec** 服务器都不再列出。参考 20.9 节了解有关安全的知识。

在编辑好/etc/inetd.conf 之后，要向 **inetd** 发一个 HUP 信号，告诉它重新读取它的配置文件，并落实您所做的任何修改。在发出信号之后，等上一会儿，然后检查日志问题，找找看有没有与修改相关的出错信息（**inetd** 把错误记录在 syslog 的“daemon”设备下）。测试所有新加入的服务，保证它们都能正确工作。参考第 10 章了解有关 syslog 的更多知识。

29.3.3 services 文件

向 **inetd.conf** 或者 **xinetd.conf** 添加了一项新服务后，可能还需要在/etc/services 文件中为其设置一项。这个文件供几个标准库使用，这些库建立了服务与端口号之间的映射关系。**xinetd** 实际上允许直接指定端口号，但是在 **services** 文件里维护一份端口的主控列表始终是一个好想法。

例如，当您键入以下命令时：

```
$ telnet anchor smtp
```

telnet 就在 **services** 文件中查找“smtp”服务的端口号。大多数系统在发售时就已配置好了所有常见的服务，如果您要添加新服务，只需编辑 **services** 文件。

[2] 如果您没有在用 **tcpd** 来保护一项服务，那么这个守护进程的第一个命令行参数应该总是守护进程自己的简写名称。这种要求并不是 **inetd** 的特殊之处，而是传统的 UNIX 习惯，这个习惯在正常情况下被 shell 掩盖了。

www.TopSage.com

TopSage.com

services 文件仅仅用于真正的 TCP/IP 服务，RPC 服务的相似信息保存在另一个配置文件 **/etc/rpc** 中。

下面是从一个 **services** 文件中选出的几行（原始文件大约有 570 行那么长）：

```
tcpmux    1/tcp                 # TCP port multiplexer
echo      7/tcp
echo      7/udp
...
ssh       22/tcp                #SSH Remote Login Protocol
ssh       22/udp                #SSH Remote Login Protocol
smtp      25/tcp    mail
rlp       39/udp    resource    # resource location
name      42/tcp                # IEN 116
domain    53/tcp                # name-domain server
domain    53/udp
...
```

每行的格式是：

```
name     port/proto aliases        # comment
```

服务通常以数字顺序列出（尽管并不要求这样的顺序）。*name* 是服务的符号名（您在 **inetd.conf** 或者 **xinetd.conf** 中使用的名字）。*port* 是服务正常监听的端口号，如果这个服务是由 **inetd** 管理的，那么这就是 **inetd** 将监听的端口[3]。

proto 指出了服务使用的协议，实际中，它一定是 tcp 或者 udp。如果一项服务既能使用 TCP 也能使用 UDP，那么每一种协议都必须有一行（如上例中的 ssh 服务）。*alias* 包含了服务的另外的名字（例如，也能够以 nicname 查找 whois）。

29.3.4 portmap：把 RPC 服务映射到 TCP 和 UDP 端口

portmap 把 RPC 服务号映射到服务器所监听的 TCP/IP 端口上。当 RPC 服务器启动时，它向 **portmap** 注册自己，列出它所支持的服务和可能连接它的端口。客户通过查询 **portmap** 找出怎样与适当的服务器连接。

这种系统允许把端口映射到符号式的服务名上。这基本上是 **services** 文件之上的另一层抽象，但是它虽然引入了额外的复杂性（以及安全问题），却没有解决任何现实问题。

如果守护进程 **portmap** “死掉”，那么依赖于它的所有服务（包括 **inetd** 和 NFS）必须重启。从实际角度看，这意味着应该重启系统。为了让 **inetd** 正确处理 RPC 服务，必须在 **inetd** 之前启动 **portmap**。

29.4 内核的守护进程

出于体系结构方面的原因，Linux 内核有几部分是按照类似于用户进程那样来管理的。这些进程通常可以由它们很低的 PID 号和以 k 开头的名字辨别出来，比如 **kupdate**、**kswapd**、**keventd** 和 **kapm**。2.6 版内核下的名字不一定这么叫，但是 ps 总是用中括号把内核线程的名字给括起来。

这些进程大多处理 I/O、内存管理和磁盘缓存同步的各方面事务。系统管理员不能操控它们，也

[3] 端口号不是随意定的。所有主机必须统一哪个服务使用哪个端口，否则，请求一定会被指向错误的端口。如果您创建了站点所特有的服务，那么应当选一个没有列入 **services** 文件的高端口号（大于 1023）。

www.TopSage.com

不应该去管它们[4]。

表 29.2 简要总结了当前内核里的主要守护进程。名字中带有一个 N 参数的守护进程（按 **ps** 命令的输出显示）在多 CPU 系统的一个 CPU 上独立运行，N 表明哪个副本在哪个 CPU 上。

表 29.2 主要的内核守护进程（2.6 版内核）

守护进程	功能	守护进程	功能
ksoftirqd/*N*	在负载重时处理软件中断	**ata/***N*	处理串行 ATA
kacpid	处理 ACPI 子系统	**scsi_eh_***N*	执行 SCSI 出错处理
kblockd/*N*	块读写子系统	**kjournald**	支持日志文件系统
aio/*N*	重试异步 I/O	**events/***N*	通用任务队列处理
kswapd/*N*	把内存页移到交换区		

在这类进程中还有一个系统守护进程，但它的名字不标准，叫做 **mdrecoveryd**。它是“多设备（译者注：在 2.0 版和 2.2 内核早期版本中，叫做 multiple devices，简称 md；在之后的内核版本中往往被称为 raid）”的一部分，这种实现更常用的名字叫 RAID。

klogd：读取内核消息

klogd 负责从内核的消息缓冲里读取日志记录项，然后把它们转发给 syslog，从而可以把它们路由到它们的最终目的地。它还能处理自身的消息，如果这么配置了的话。参考 10.2.4 节了解更多有关内核和引导时刻日志记录的资料。

29.5 打印守护进程

常用的打印系统有好几种，每一种都有自己的一套命令和守护进程，提供和打印相关的服务。在有些场合中，各套系统被组合使用；而在另一些场合中，在一个系统上有多套不同的系统。

29.5.1 cupsd：CUPS 的调度器

CUPS（Common UNIX Printing System，通用 UNIX 打印系统）提供了一种可移植的打印机制，它是按照 Internet 打印协议（Internet Printing Protocol）1.1 版实现的。它允许远程用户用 Web 界面向其办公室打印资料（反之亦然）。CUPS 已经相当流行，成为大多数系统上默认的打印管理程序。它也很灵活，能做远程的身份验证。

29.5.2 lpd：管理打印

lpd 负责老式的 BSD 假脱机打印系统。它从用户那里接受打印作业，然后派生出进程执行实际的打印操作。**lpd** 也负责在远程系统间来回传递打印作业。有时候 **lpd** 会挂起，此时必须手工重新启动。

您的系统可能是原来的 **lpd** 形式，或者是特别好的版本（这是 LPRng 软件包的一部分）。参考第 23 章，了解有关这些打印软件的更多资料。

[4] 如果您熟悉内核的实现，那么改变这些进程执行的优先级偶然会有用处。不过，这不是标准的系统管理任务。

www.TopSage.com

29.6 文件服务守护进程

下面的几个守护进程是 NFS 文件共享系统的一部分。我们在这里只简要介绍它们的功能，第 16 章已对它们做了详细说明。

29.6.1 rpc.nfsd：提供文件服务

rpc.nfsd 运行在文件服务器上，处理 NFS 客户端的请求。在大多数 NFS 的实现里，**nfsd** 实际上都只是内核的一部分，但出于调度方面的原因而被伪装成了一个进程的样子。Linux 实际上有两种不同的实现，一种符合这种习惯，而另一种则在用户空间运行。采用内核实现的方式更普遍，也是大多数发行版本默认的选择。

rpc.nfsd 接受一个参数，这个参数指定了能派生出多少个它自己的副本。在选择正确的副本数量时蕴涵着一些技巧，参见 16.2.2 节的内容。

29.6.2 rpc.mountd：响应安装请求ⓘ

rpc.mountd 接受从 NFS 客户端来的文件系统安装请求。它验证每个客户端是否有权安装它所请求的目录。**rpc.mountd** 查阅**/var/state/nfs/xtab** 文件来判断哪些申请者是合法的。

29.6.3 amd 和 automount：根据需要安装文件系统

amd 和 **automount** 是 NFS 自动安装程序，它们是守护进程，等到进程要用一个文件系统的时候，它们就会实际安装这个文件系统。如果在一定时间内没有进程访问这些文件系统，自动安装程序随后就卸载它们。

在一个大型环境中，要在网络上共享数十乃至数百文件系统的时候，使用自动安装程序非常有用。因为网络上所有的系统都共享相同的 **amd** 或者 **automountd** 的配置，所以自动安装程序提高了网络的稳定性，降低了配置的复杂性。我们在 16.6 节详细介绍了如何使用标准的 Linux 自动安装程序。

29.6.4 rpc.lockd 和 rpc.statd：管理 NFS 锁

虽然 **rpc.lockd** 和 **rpc.statd** 是两个不同的守护进程，但它们总是一起运行的。**rpc.lockd** 维护着 NFS 文件上的建议性锁（advisory lock，按 flock 和 lockf 的方式使用）。**rpc.statd** 能让进程监视其他正在运行 NFS 的机器的状态。**rpc.locked** 使用 **rpc.statd** 来决定何时尝试与远程机器进行通信。

29.6.5 rpciod：缓存 NFS 数据块

rpciod 缓存 NFS 客户机的读写请求。它执行“先读”和“后写”两种缓冲机制，大大提高了 NFS 的性能。这个守护进程类似于其他系统上的 **biod** 和 **nfsiod** 守护进程，虽然它在结构上有点不同。

www.TopSage.com

29.6.6 rpc.rquotad：远程磁盘配额服务

rpc.rquotad 让远程用户检查在他们已经用 NFS 安装的文件系统上自己有多少配额。配额限制实际上仍然是在服务器上实现的，**rpc.rquotad** 只是让 **quota** 命令能够正确工作。

29.6.7 smbd：向 Windows 客户端提供文件和打印服务

smbd 是 Samba 软件包中的文件和打印机服务器。它通过以前陆续叫做 SMB 或者 CIFS 的 Windows 协议提供共享文件和打印机的服务。参见 26.6 节的内容了解详情。

29.6.8 nmbd：NetBIOS 名字服务器

nmbd 是 Samba 的另一个组成部分。它回答 Windows 机器发出的 NetBIOS 名字服务请求。它还实现了一种浏览协议，Windows 用它构成“我的网络（My Network Places）”文件夹，并使本地主机上的共享磁盘能从那里对外界可见。**nmbd** 也能当作一台 WINS 服务器使用。

29.7 管理数据库守护进程

本书的第 17 章介绍了几种与 Sun 的 NIS 管理数据库系统有关的守护进程。虽然 NIS 最早来源于 Sun 公司，但是现在它也在包括 Linux 在内的其他厂商的系统上使用。

29.7.1 ypbind：定位 NIS 服务器

在所有的 NIS 客户机和服务器上都要运行 **ypbind** 守护进程。它找到一台能向其发查询请求的 NIS 服务器。**ypbind** 实际上并不处理请求本身，它只是告诉客户程序用哪个服务器。

29.7.2 ypserv：NIS 服务器

在所有的 NIS 服务器上都要运行 **ypserv**。**ypserv** 接受从客户机来的请求，并且把被请求的信息回复给它们。参考 17.4.5 节了解如何配置运行 **ypserv** 的机器。

29.7.3 rpc.ypxfrd：传送 NIS 数据库

rpc.ypxfrd 以一种高效的方式把 NIS 数据库传送给从服务器。从服务器用 **ypxfr** 命令发起传送。只要主服务器上的数据库发生了变化，就应该立即把它“推”送给所有的从服务器，从而保证 NIS 服务器彼此一致。

29.7.4 lwresd：轻量级名字解析器库服务器

lwresd 提供了一种快速的方法来缓存“地址到主机名”和“主机名到地址”查询的结果。它供解析器（系统标准库的一部分）访问，并且许多程序也直接调用它。这个库和守护进程通过一种简单的

www.TopSage.com

UDP 协议进行通信。

29.7.5 nscd：名字服务高速缓存守护进程

nscd 缓存调用 **getpw***、**getgr*** 和 **gethost*** 系列的标准 C 库函数的结果，这些库函数查询的数据传统上都保存在 **passwd**、**group** 和 **hosts** 文件里。现如今，数据的可能来源范围比原来大，包含了像 NIS 和 DNS 这样的候选来源。**nscd** 实际上不知道数据是从哪儿来的，它只是把结果缓存下来，后续的库调用就可以走捷径使用这些结果了。高速缓存的策略在**/etc/nscd.conf** 文件里设置。

29.8 电子邮件守护进程

除了得到广泛使用的核心的 sendmail 和 Postfix 邮件投递系统之外，还有几个守护进程有助于从远程访问邮箱。

29.8.1 sendmail：传输电子邮件ⓘ

sendmail 的任务包括接收来自用户和远程站点的消息、重写地址、扩展别名以及在 Internet 上传送邮件。**sendmail** 是一种重要并且非常复杂的守护进程。参考第 18 章了解有关它的完整知识。

29.8.2 smtpd：简单邮件传输协议守护进程ⓘ

smtpd 监听传入到端口 25 上的电子邮件消息，并把它们转给后端的传输系统做进一步的处理。参考 18.3.1 节和 18.16.2 节了解更多关于在 **sendmail** 和 Postfix 系统上使用 **smtpd** 的知识。

29.8.3 popd：基本的邮箱服务器ⓘ

popd 守护进程实现了 POP 协议（Post Office Protocol，邮局协议）。非 Linux 系统通常使用该协议来接收电子邮件。

29.8.4 imapd：高级的邮箱服务器ⓘ

imapd 守护进程实现了 IMAP 协议（Internet Mail Access Protocol，Internet 邮件访问协议），它是替代 POP 的更高级和功能更强的协议。它允许基于 PC 的用户（或者有启用了 IMAP 功能的邮件阅读器的 Linux 用户）从各种不同的地方访问自己的邮件，邮件文件夹则保存在 Linux 服务器上。关于 IMAP 的更多信息请访问 www.imap.org 这个站点。

29.9 远程登录和命令执行守护进程

通过网络登录到系统中并且执行命令，这是开发 UNIX 网络功能最早的动机之一，这个功能仍然是现如今系统管理工作的基本内容之一。遗憾的是，UNIX 界耗费了数十年时间才让这种技术在安全上面达到成熟。现代的生产系统都应该使用 SSH（**sshd**）而再用不着别的。

www.TopSage.com

大家网 TopSage.com

29.9.1 sshd：安全的远程登录服务器ⓘ

sshd 提供的服务与 **in.rlogind** 相似，但它的会话是通过一条加密通道来传输（和验证）的。加密算法各种各样。因为当今的 Internet 环境相当险恶，所以您必须只允许通过像这样的守护进程——而非 **in.rlogind** 或者 **in.telnetd**——从 Internet 上访问 shell。在 20.11.3 节可以找到有关 **sshd** 的更多信息。

29.9.2 in.rlogind：过时的远程登录服务器ⓘ

in.rlogind 是很久以前负责处理远程登录的标准进程。当它由 **inetd** 调用时，它尝试通过检查系统级的**/etc/hosts.equiv** 文件和本地用户的**~/.rhosts** 文件自动验证远程用户的身份。如果自动身份验证成功，那么用户就可以直接登录。否则，**in.rlogind** 执行 **login** 程序提示用户输入口令。因为它的身份验证相当简单，所以 **in.rlogind** 是一种安全隐患。关于这个主题的更多信息请参见 20.9.3 节。

29.9.3 in.telnetd：另一种远程登录服务器ⓘ

除了使用 TELNET 协议之外，**in.telnetd** 与 **in.rlogind** 非常相似。TELNET 协议允许双方（客户机和服务器）协商流控和双工的设定，因此与 **in.rlogind** 相比，这个协议对于慢速的或者不可靠的连接而言是一种更好的选择。与 **rlogin** 相似，**telnetd** 在网络上传输明文口令。因此在现代网络中不鼓励使用它。不过，许多非 Linux 的系统都支持 **telnet**。

29.9.4 in.rshd：远程命令执行服务器ⓘ

in.rshd 处理来自 **rsh** 和 **rcmd** 的远程命令执行请求。**in.rshd** 采取的身份验证过程与 **in.rlogind** 的相似，只是处理自动身份验证不能通过时的情况不同，此时 **rshd** 不会让用户提供口令，而是拒绝用户的请求。**in.rshd** 也是用于 **rcp**（远程 **cp**）的服务器。和 **in.rlogind** 一样，**in.rshd** 也在最近因为安全原因而成了众矢之的，一贯地遭到禁用。参见 20.9.1 节了解更多信息。

29.10 引导和配置守护进程

在 20 世纪 80 年代，UNIX 世界掀起了一股无盘工作站的热潮。这些机器完全通过网络引导，并通过 NFS 这样的远程文件系统技术来执行所有磁盘操作。随着磁盘价格的下跌以及速度的提高，人们对无盘工作站的兴趣很快就消失了。然而，它们可能随时会像 20 世纪 70 年代的“松糕鞋”那样又重新流行起来。无盘时代的两个主要遗留物是：为支持无盘系统而设计的大量守护进程，以及大多数厂商的文件系统都有奇特的组织结构。

出于好奇，我们在 11.2 节稍微详细地讨论了一下无盘系统本身。

虽然无盘工作站不再常用，但是它们的引导协议却被其他设备“霸占”了。大多数可管理的网络集线器和网络打印机都使用本小节列出的几种服务的组合来进行引导。

29.10.1 dhcpd：动态地址分配ⓘ

在引导系统时，DHCP（Dynamic Host Configuration Protocol，动态主机配置协议）给 PC、膝上

www.TopSage.com

机以及其他“移动”平台提供了关于它们的 IP 地址、默认网关以及名字服务器的信息。**dhcpd** 是在 Linux 下实现这种服务的守护进程。在 12.9 节能够找到关于 DHCP 的更多信息。DHCP 更精致的版本叫做 PXE（Pre-boot eXecution Environment，引导前执行环境），它能帮助兼容它的机器从网络引导，而且不需要本地引导设备，参见 21.1.2 节了解详情。

29.10.2 in.tftpd：简单文件传送服务器ⓘ

in.tftpd 实现了类似于 **ftpd** 的一种文件传输协议，但是要简单得多。许多无盘系统使用 TFTP 从一台服务器下载它们的内核。**in.ftpd** 不执行身份验证，但是它在正常情况下只限于提供一个目录里的文件服务（通常是/**tftpboot**）。因为 TFTP 目录下放的任何内容都能在整个网络上访问到，所以这个目录应该只包含引导文件，不应该公共可写。

29.10.3 rpc.bootparamd：高级的无盘支持ⓘ

rpc.bootparamd 使用文件/**etc/bootparams** 告诉无盘客户机从哪儿能找到它们的文件系统。使用 RARP 得到自己的 IP 地址的机器，以及使用 NFS 来安装自己的文件系统的机器经常会用 **rpc.bootparamd** 服务。

29.10.4 hald：硬件抽象层（HAL）守护进程

hald 从多个来源收集有关系统硬件的信息。它通过 D-BUS 提供一个活的设备列表。

29.10.5 udevd：设备连接串行化通报

udevd 是 udev 动态设备命名系统的一个不大的组成部分。它考虑到了热插拔事件的正确次序，内核有时候可以和用户空间不按顺序进行通信。

29.11 其他网络守护进程

下面的守护进程使用 Internet 协议来处理请求。这些“Internet”守护进程中有许多实际上把大部分时间花在处理本地请求上。

29.11.1 talkd：网络聊天服务ⓘ

从 **talk** 程序来的连接请求是由 **talkd** 来处理的。当 **talkd** 接收到一个请求时，它就与其他机器进行协商，在两个已经执行了 **talk** 的用户之间建立一条网络连接。

29.11.2 snmpd：提供远程网络管理服务

snmpd 对使用 SNMP（Simple Network Management Protocol，简单网络管理协议）的请求做出响应。SNMP 对一些常用的管理操作进行了标准化。关于 SNMP 的更多信息请参见 19.8 节。

www.TopSage.com

29.11.3 ftpd：文件传输服务器ⓘ

守护进程 **ftpd** 处理来自 **ftp**（Internet 文件传输）程序的请求。许多站点都禁用它，不是因为它耗费大量资源，就是因为那些站点担心安全问题。**ftpd** 可以被设置成允许任何人从您的机器上传走或向您的机器传入文件。关于 **ftpd** 的更多信息请参见 21.6 节。

Linux 系统上有很多不同的 **ftpd** 的实现。如果您打算运行一个流量很大的服务器，或者需要像负载管理这样的高级功能，那么最好研究一下能替代 Linux 发行版本上默认 **ftpd** 的其他软件。

华盛顿大学（Washington University）开发的 WU-FTPD 是最流行的标准 **ftpd** 替代软件。访问 www.wu-ftpd.org 了解更多信息。

29.11.4 rsyncd：在多台主机间保持文件同步ⓘ

rsyncd 实际上只是 rsync 的一个链接而已，选项**--daemon** 把它变成了一个服务器进程。**rsyncd** 使得文件在多台主机之前更容易保持同步。它本质上是 **rcp** 的一个高效且考虑到安全的版本。对于系统管理员来说，**rsync** 可是个宝，我们在这本书里已经在两个不同的地方介绍了它的使用。参见 17.3.2 节了解使用 **rsync** 共享系统文件的一般信息和一些技巧。**rsync** 也是许多站点的内部安装过程的一个重要部分。

29.11.5 routed：维护路由表

routed 维护着供 TCP/IP 在网络上发送和转发分组的路由信息。**routed** 仅仅处理动态路由信息，**routed** 从来不改变静态定义的路由（也就是用 **route** 命令配到系统路由表中的路由）。相对来说，**routed** 比较笨、效率低，我们建议只在几种特殊的情况下使用它。关于 **routed** 的更详细的讨论参见 13.4 节。

29.11.6 gated：维护复杂的路由表

gated 理解几种路由协议，其中包括由 **routed** 使用的 RIP 协议。**gated** 在不同的协议之间转换路由信息，它的可配置性很高。与 **routed** 相比，它对于网络而言更友善。关于 **gated** 的更多信息请参见 13.5 节。

29.11.7 named：DNS 服务器

named 是最为流行的域名系统服务器。它使用由各地的 **named** 所维护的分布式数据库将主机名映射到网络地址上，并且执行许多其他功能。第 15 章描述了 **named** 的有关内容。

29.11.8 syslogd：处理日志消息

syslogd 充当了清理系统软件和守护进程产生的状态信息和错误消息的角色。在 **syslogd** 被编写出来之前，守护进程不是把错误消息直接写到系统控制台上，就是保留它们自己的专用日志文件。现在，守护进程使用 **syslog** 库例程把消息传输到 **syslogd**，**syslogd** 根据系统管理员建立的规则对消息进行分类。关于 **syslog** 的更多信息请参见 10.4 节。

www.TopSage.com

29.11.9 in.fingerd：查找用户ⓘ

in.fingerd 提供有关登录到系统中的用户的信息。如果查询 **finger** 的话，它也能够提供各个用户的更多细节。实际上，**in.fingerd** 本身并没做很多工作：它仅仅接受输入行并把它们传送给本地的 **finger** 程序。

finger 能够返回关于用户的不少信息，包括用户的登录状态，用户在**/etc/passwd** 文件中 GECOS 字段的内容，以及用户的**~/.plan** 文件和**~/.project** 文件的内容。

如果您连入了 Internet 并且正在运行 **in.fingerd**，那么世界上的任何人都能够获得这个信息。**in.fingerd** 使人们得到了一些真正需要的服务（例如 Internet 白页），但也能够使人们做各种坏事（比如找人对其进行电话骚扰，以及查找能发垃圾邮件的地址）。一些站点通过关闭 **in.fingerd** 来反击这种入侵，而其他站点仅仅是限制 **in.fingerd** 返回的信息量。不要想当然地认为因为 **in.fingerd** 简单，那么它就应该安全——在 1988 年发生的最早的 Internet 蠕虫就是针对 **in.fingerd** 的缓冲区溢出进行攻击的。

29.11.10 httpd：World Wide Web 服务器ⓘ

httpd 使您自己的站点变成了 Web 服务器。**httpd** 能够向其客户机发送文本、图片和声音。关于提供 Web 网页服务的更多信息请参见第 21 章。

29.12 NTPD：时间同步守护进程

随着计算机之间相互依赖性的日益增强，对计算机而言，共享一致的时间也变得越来越重要。在发生安全入侵的情况下，同步的时钟对于把日志文件项关联起来是很重要的，而且它们对于许多终端用户应用（从软件项目的联合开发到金融交易的处理）也非常重要。

ntpd[5]实现了网络时间协议（Network Time Protocol，NTP），该协议能让计算机彼此把自己的时钟同步到几毫秒的精度内。第一个 NTP 实现在 1980 年前后启动，准确度只能达到数百毫秒。现如今，一种新的内核时钟模型能让时间的精度达到 1 纳秒。最新版的 NTP 协议（第 4 版，在 RFC 2783 中说明）保持和以前版本的兼容性，增加了方便的配置和一些安全特性。

NTP 服务器分级部署，每级叫做一个“stratum（层）”。在 Stratum-1 上的服务器通常要依赖一个外部参考时钟，比如无线接收器或者原子钟。Stratum-2 的服务器按 Stratum-1 的服务器来设自己的时钟，它们起到了时间发布中心的作用。Stratum 多达 16 层。一个时间服务器要确定自己的 stratum，只要把它去同步的服务器的最高 stratum 数加 1 即可。1999 年由 Nelson Minar 做的一份 NTP 网络调查指出，stratum-1（当时）有大约 300 台服务器，Stratum-2 有 20 000 台服务器，Stratum-3 则超过了 80 000 台。

现如今，NTP 客户机能够访问到若干种参考时间的标准，比如有 WWV 和 GPS 提供的那些时间标准。

大多数 ISP 都维护着自己的一组时间服务器，它们应该从网络角度靠近下游的客户（如果 NTP 正确工作的话，就会准确）。

ntpd 实现了 NTP 协议的客户端和服务器端。它在启动时读取**/etc/ntp.conf** 文件。在这个配置文件中，您可以指定访问、客户机网络、时间服务器、多播客户机、一般配置和身份验证方面的配置，但

[5] 这个守护进程较早的实现版本也叫做 **xntpd**。

www.TopSage.com

是别被吓住——配置都很直观明了。

Debian 和 Ubuntu 默认似乎没有带 **ntpd**，但通过 **apt-get** 就能得到它。您还可以从 ntp.isc.org 获得该软件的当前版本。

您也可以使用简便的 **ntpdate** 工具用一台 NTP 服务器设置系统的时钟。这种方案没有 **ntpd** 好，因为它让时间变得不连续了。突然把时钟往回调特别有害，因为程序有时候假定时间都是单调增加的函数。

ntpd 使用系统调用 **adjtimex** 来使系统时钟的调整更平稳，并且避免向前或者往回有很大的时间跳跃。**adjtimex** 调整系统的时钟速度，让它逐渐校准。当系统的时间和正确的目标时间相符合的时候，就取消调整，时钟正常运行。

29.13 习题

★E29.1 使用 **ps** 判断在您的 Linux 系统上正在运行哪些守护进程。另外判断通过 **inetd.conf** 或者 **xinetd.conf** 可以运行哪些守护进程。把这两个问题的答案结合起来，说明每个守护进程能做什么，它在哪儿启动，是否能（或者在）同时运行多个副本，以及您可以从 **inetd.conf** 和 **ps** 的输出中了解到的其他任何属性。

★E29.2 在实验室里，启用网络时间守护进程 **ntpd**。

a）怎样能够辨别出您的系统有了正确时间？

b）用 **date** 命令手工把您的系统时间调慢 15 秒。它要（或者将会）花多长时间才能让时间恢复正确值？

c）手工把您的系统时间调快一个月。**ntpd** 对这种情形会产生什么样的反应？（要求有 root 权限）

★★E29.3 在实验室里，使用诸如 **netstat** 这样的工具来判断机器上有哪些端口处于“监听”状态。

a）您怎样把 **netstat** 信息和 **inetd.conf** 中找到的信息协调起来？如果有差异，该怎么办？

b）在另一台不同的机器上安装 **nmap** 工具。运行针对您的系统的端口扫描，验证从 a）了解到的信息。（如果有的话）从 **nmap** 又了解到了什么从 **netstat** 看不明显的信息？（参见 20.10.1 节了解有关 **nmap** 的更多知识。）

www.TopSage.com

大家网
TopSage.com

第 30 章　管理政策与策略



您可能有着最聪明的系统管理员团队，但是如果您的技术管理工作不足，那么您不会喜欢这样的生活，而您的用户也会如此。本章我们主要介绍运作一个成功的 IT 支持单位在非技术方面的话题，还要介绍几个技术性的问题（基础设施设计、故障工单系统等），它们有助于支持从事系统管理工作的管理人员。

本章介绍的这些话题和思想并不特定于某个特殊的环境。无论是兼职的系统管理员，还是负责一个重要 IT 部门的全职人员团队都同样适用。和绿色蔬菜一样，无论您准备做什么菜，它们都适用。对于有着数百名 IT 雇员、确实很大的单位来说，我们也会简要介绍一种面向流程的管理系统结构——信息技术接口库[ITIL]，它主要适用于大单位。参考 30.14.3 节的内容。

除了管理之外，我们还包含进来了一些章节：专门介绍建档（30.8 节），故障工单系统（30.9 节），灾难恢复（30.10 节），书面政策（30.11 节），法律问题（30.12 节），软件专利（30.13 节），标准化工作（30.14 节），Linux 文化（30.15 节）以及和 Linux 有关的专业机构、会议和系统管理员的培训机会（30.17 节）。

30.1　皆大欢喜

系统管理是一个服务行业，用户和计算机都是服务的对象。有些更注重技术的系统管理员似乎有这样的印象，即他们所负责系统的用户都不过是妨碍他们实际工作的人。这种观点于事无补，也不准确。我们的工作就是向用户提供 IT 基础设施，用户需要这些设施高效而可靠地完成自己的工作。

在我们参加工作后都有站在咨询台外面询问别人的时候。所以您要设身处地地为您的客户着想：站在咨询台前经历的感觉是怎样的？想到这些的话，您的态度会发生怎样的变化？

www.TopSage.com

下面的情形会让用户感到高兴：

- 他们的计算机开着而且在运行，他们能登录进系统；
- 他们的数据文件保持原封不动；
- 他们的应用软件应该已经安装好了，而且能运行；
- 需要的时候能得到友好、博学的帮助。

用户希望每星期 7 天、每天 24 小时都是这样。最好能免费。用户讨厌的情形有：

- 无论有安排还是没有安排的停运；
- 升级引起了突发的不兼容变化；
- 从系统或者管理员得到不能理解的消息；
- 对为什么不能工作长篇累牍的解释。

当出现问题时，用户想知道什么时候能修好。这就够了。他们并不关心是哪块硬盘或者哪个发电机坏了，或者为什么坏了；把这些信息留给您的管理报告吧。

从用户的角度来看，没有消息就是好消息。系统要么能干活，要么不能，如果不能，为什么不能没什么关系。我们的客户最高兴的就是他们甚至注意不到我们的存在！让人难过吧，但却是真的。

用户是我们的上帝，但让您的员工心情愉快也同等重要。好的系统管理员很难得，在设计您站点上的系统管理体系的时候，就必须要考虑他们的需求。

下面的情形会让系统管理员和其他技术人员高兴：

- 他们的计算机和支持系统开着而且在运行；
- 他们有做自己工作必备的资源；
- 他们有最新和最好的软硬件工具；
- 他们的工作不会老被打断；
- 老板不管闲事和插手细节，他们就会有创造性；
- 他们的工作时间和强度在合理的范围内。

推动技术人员要的不仅是月末的工资单。他们需要感到能在一定程度上创造性地控制自己的工作，而且能得到同行、老板和用户的认可。

让客户和 IT 员工皆大欢喜的要求有某些共同的因素。不过，有些地方似乎没关系，甚至是相互矛盾的。老板必须能保证让所有这些不一样的期望能和谐共处，而且可以做到。

在有些单位，由老板和客户接触，以此保护技术人员不会在一线受到打搅，放手让他们做好自己的“真正”工作。我们的经验是，这种安排通常并不是个好的想法。它把技术人员隔离起来，不能直接接触到客户的需求，经常造成老板的承诺员工无法做到的情况出现。

30.2 IT 职能机构的组成

随着系统管理支持机构不断壮大，显然组内不会每个成员都能或者都应该掌握整个 IT 基础设施的一切。要一个人对每天所有的事情做决定也是不切实际的（的确不可能）。随着机构的扩大，会逐步形成下面这些职能部门。

- 管理部门：制定全面战略，领导整个机构。
- 行政部门：和客户交谈，了解他们的需求、谈合同、开发票、下订单、付货款、管理员工聘用和解聘的处理过程。
- 开发部门：设计、实现和测试新产品和服务，之后交付客户。
- 执行部门：直接向客户提供服务，包括系统管理、终端用户支持和培训。我们把这个领域细分为运行（和机器打交道）和支持（和人打交道）。

www.TopSage.com

这些职能部门没必要一定对应到个人。甚至在只有一个系统管理员的小运行部门，上述 4 个领域也是这个系统管理员所必须扮演的不同角色。

每个部门的相对规模反映出系统管理团队所服务的单位类型。其组成在一所研究型大学与一家软件开发公司或者制造公司也不一样。我们在研究型大学取得的经验表明，管理大约占 5%的工作量，行政占大约 10%，开发占 10%～20%，而执行占其余的 65%～75%。

我们建议，在中小型单位里，所有的雇员都至少要在支持部门做兼职。这个习惯能够让技术人员在问题、挫折和客户要求上面获得第一手的认识。随后可以把客户反馈用于改进当前的环境，或者设计未来的系统和服务。还有一项作用，就是会让客户接触到一些知识更丰富的雇员（陷入一线支持的工作拔不出来了？）。

开发占 20%显得有点儿高，但是没有开发，您的解决方案就不能随着被支持系统数量和复杂性的增加而扩展。您原来的管理思想就会没有发展，最终过时而没有补救的希望。

一个人直接管理的人员数量介于 5～10 个之间。超过这个数量的话，就要考虑增加更多的管理级别。

我们在下面的几小节中讨论这几个方面的任务。

30.3 管理的职能

经理的全部目标就是促进技术人员的工作，这些人是单位里做“实际”工作的专家。下面几项任务和职责落在了经理的肩上：

- 领导团队、指引方向、提供必要的资源；
- 聘用、解聘、人员评估和技能培养；
- 给员工分配任务，跟踪进度；
- 管理工作；
- 处理问题：人员矛盾、难缠用户、老旧硬件等；
- 充当用户向上反映问题的“上级领导”；
- 监督可扩展基础设施的发展；
- 应对灾难和紧急事件的规划；
- 让古怪的系统管理员整理出文档。

这份清单里似乎把和客户有关的任务都省略了。不过，我们认为，这个职务实际上最好由技术人员担任。经理通常不要求具有技术背景去评估客户要求的难度和可行性。当承诺给客户的产品和进度由完成实际工作的那些人参与决定时，双方就没什么疑问了。

下面我们将更细致地讨论这些管理职责中的一些。其中有些职能跨越了多个职能领域，在别的部门里也有。例如，开发一个可扩展的基础设施就是由管理部门和开发部门联合承担的项目，参考 30.5 节。类似地，灾难恢复也既是管理任务，也是运行任务，我们在 30.6 节讨论。建立文档很重要也很难，所以由专门的 30.8 节介绍。

30.3.1 领导

领导工作不容易说清。但是领导力不足或者领导得不好时，缺乏领导的问题就很容易暴露出来。从某种方式看，领导工作就是单位的“系统管理”，领导要确定方向、保证下级团结合作，而且要让整个系统运行时尽可能没有错误。

遗憾的是，将某个人造就为一名伟大的计算机系统管理员的技术绝活未必能转变为领导艺术，后

www.TopSage.com

者要求更多的是一套以人为核心的能力。人要比 Perl 难掌握得多。

对于有很强技术背景的新经理来说，特别难把重心放到管理工作上，避免情不自禁地一头扎到工程工作上去。沉醉于解决一个技术问题要比和“难对付的”员工长时间谈话更舒服，也更有意思。但是后者对单位更有价值。

一种检查您的领导水平的简单（或许让人吃惊）方法如下。写下您单位里正在进行的任务单。用一种颜色标出您在其中起掌控作用的领域，用另一种不同的颜色标出您在其中具体干活的领域。哪种颜色占多数？

30.3.2 聘用、解聘和人事管理

人事管理特别富有挑战性。作为您监管职责的一部分，您要和雇员的技术和个性两个方面打交道。正如人们一贯认为的那样，技术上出色的系统管理员往往不善沟通，他们有时候显得和机器而不是人相处得更好。作为他们的经理，您需要让他们在两方面都提高。

技术的提高相对容易推动和量化，而个性的提高也同等重要。下面是在评估雇员的对外交流能力上要问的一些重要问题。

- 这个人的举止适合您的工作环境吗？
- 这个人和上级、客户以及供货人的关系如何？
- 这个人能和团队内的其他成员搞好关系吗？
- 这个人对批评和技术争议的态度是怎样的？
- 这个人能主动克服技术上的不足吗？
- 这个人的沟通能力如何？
- 这个人能计划、实施和演示一个客户的项目吗？

一、聘用

重要的是，对有可能成为新雇员的人以及现有的老雇员都要做这些评估。申请职位者的个人素质经常被忽略或者低估。在这个方面不要投机——您以后肯定会后悔的！

一次个人面试可能可以回答其中的一些问题。而一次和证明人的电话谈话通常能告诉您更多的信息。要非常仔细地听，许多人不喜欢说前任雇员或者同事的任何坏话，所以他们会找出更巧妙的方式告诉您（如果您仔细倾听的话）那个雇员有问题。如果申请者没有把最近的雇主作为证明人，那么就非常值得怀疑。

成立系统管理员队伍有两种方法：

- 聘请有经验的人员；
- 自己培养系统管理员。

有经验的管理人员通常能够更快地适应工作，但您总是会不想让他们了解某些事情。系统管理工作要求必须有 root 访问权限。但由于对所雇的系统管理员不十分了解，因此您可能不愿意让他们立即掌握公司的数据。随着您对他们信任的加深，让他们介入可能要多做一些监视和审计。

系统管理员的培训需要花费相当的时间和精力，同时一个投入运行的网络也不是理想的培训场所。但如果人员筛选得当（具有反应敏捷、兴趣广泛、求知欲强、细心等优点），往往会收到更好的效果。

在一名好的系统管理员身上，可能有几样素质是相互矛盾的。一名系统管理员在处理某个问题时必须勇于尝试创新的解决方案，但又必须足够细心不会尝试做任何有破坏性的事情。人际交往和解决问题的能力都很重要，而在我们所认识的许多系统管理员身上，这两种能力似乎没有什么关系。我们的审稿人之一说“有风度的系统管理员”是有矛盾的修辞手法。

我们已经开发了两种工具用于对有经验的申请人进行水平评估，我们过去称其为“测试”，但发

www.TopSage.com

现有些机构（像美国的州立大学）不允许对申请人进行测试。因此我们不再进行测试，而只是进行评估。

第一项是书面评估，即要求申请人对自己的经验和对不同系统及网络任务的了解程度进行评价。熟悉程度按照0～4的等级划分：

- 以前从未听说过（0）；
- 听说过，但没有具体操作过（1）；
- 在其他人的监督下操作过（2）；
- 能够独立操作，无需其他人的监督（3）；
- 十分了解并可对其他人进行指导（4）。

这些问题中暗藏许多机关。例如，在硬件部分是一个关于RJ-45连接器的问题，后面跟着是一个“MX连接器（MX指DNS里的邮件交换记录，而不是一个网络或者串口连接器）”的问题。通过这些隐蔽的问题即可对申请人回答中的瞎蒙因素做出评估。对MX连接器那个问题的回答为3的则要怀疑。结束上述书面评估之后，您可以天真地问：“那么您使用MX连接器的目的是什么呢？”

第二项评估专门用于电话面试。问题是这样提的，对于了解情况的申请者来说，可以很快地给出这些问题的答案。对正确的答案我们计+1分，回答不知道则计0分，对明显的瞎蒙或在背后键入man xxx得到答案的则计–1分。对于“我不知道，但是我可以这样找出答案。”这样形式的合理答案，也可以计+1分。知道到哪儿查往往就等于心里记得详细。

对于我们来说，这两种方法效果都相当不错。我们使用隐藏问题的比例是由我们的人力资源部门的人来确定的，一两个问题不足以说明某个人不合格。

二、解聘

如果您在用人方面出现了错误，则应及早纠正。您可能会错过一些大器晚成的人，而选择了过于自负的家伙，这些人在上任之后将排除异己，使系统管理组的人员变得越来越疏远。您的客户也会发现“这样那样”没有解决的问题，而且开始要求一个特定的系统管理员为他们工作。您不需要客户来插手您日常工作里做出的管理决定。

在许多单位里，解雇人是非常困难的事情，尤其是在试用期结束后。因此要确保试用期的评估十分严格。而后您可能还需要收集显示其缺乏能力的数据，给予正式的警告，制定表现目标等。在极端的情况下，解聘一个不称职但又不缺能力的雇员，唯一的方式就是裁掉他的工作岗位。

三、彻底测试、保证质量

经理给怎样算完成任务定下了基调。没有经验的系统管理员常常认为某个问题已经解决，而将若干更严重的故障报告只看作是一些慢慢去彻底、准确完成的任务。这种情况之所以出现，是因为起初报告问题的用户没有描述清楚，或者可能因为用户提出了一个不充分（或者不正确）的解决方案，而系统管理员没有不怕麻烦地做一次独立的诊断，就欣然接受了。同样可能的原因是系统管理员没有认真测试过解决方案。

下面是一些常见的错误：

- 没有为新软件安装手册页和相关文档；
- 软件没有在所有机器上都安装；
- 软件只由系统管理员所有，或者安装的权限是错误的。

测试是累人的，但繁忙的系统管理员跳过不做又会使软件的工效降低一半。无论是用户还是系统管理员都需要对故障报告投入时间和精力[1]。只有当所有的操作问题都被找到并解决以后，工作才算完成。

[1] 有时候，开一张故障工单或者打一次咨询电话都是硬性成本。例如，外包的支持目前大约是每个电话或者每个新故障工单75美元。

www.TopSage.com

一个常见的问题是有用户报告说“X 无法在机器 Y 上工作”，接着系统管理员到机器 Y 上尝试使用 X，这一次 X 却可以正常工作。对故障报告回答“我就能用”的说法就多少带有一点自己的态度。

如果系统管理员以提交报告的用户的身份对命令进行检测（例如在命令前运行 **sudo su - username**），就可能让问题一下子暴露出来。**su** 中“-”参数的作用是使产生的 shell 使用 **su** 成的那个用户的环境。由此可知，您确实可以对出现故障的环境进行模拟。

有时候要给测试制定个方案。例如，如果您怀疑问题 P 是由 X 引起的，那么删除 X，检查问题 P 是否没有了。然后再重新安装 X，看看 P 有没有再出现。最后，删除 X 核实 P 解决了。这个过程让您有两次机会注意到 P 和 X 实际上是否没有关系。

如果系统管理员第一次尝试却没有排除故障，将会使用户感到很生气。因此应当让用户具有适当的期望度。让报告问题的用户和您一起解决问题往往非常有用，尤其是当故障出现在不熟悉的软件包上时更是如此。这样做不但能够获得更多的信息，而且用户也不大会把同您的关系看成是对立的关系。

四、管理但别管闲事

作为一位技术能力很强的经理，您常常会情不自禁地给雇员提建议，告诉他们该怎么做自己的工作。但是如果您这么做了，您可能就没有给他们成长到可以完全对其工作负责的机会。如果您不仅提建议，而且还决定工作该怎么做，那么就更糟糕了。这样管得太细会让沮丧的员工无法发挥自己的技能和创造力。

如果您确实对每件事都比您的员工更懂行，那么要么是您的单位不对（您负责的面儿太窄），要么是您做了错误的工作，或者您有了错误的员工。否则的话，您可能只是一个爱控制的怪人。

五、表现好的奖励假期

大学老师享受休假——一年或者 6 个月，在此期间他们没有教学任务，能够到另一所大学他们个人感兴趣的新项目上做科研。这种做法的思想是，大学老师要尽力做好教学和科研工作的话，就需要接触其他人和学校。

系统管理员也需要休假，但是经常认为他们不可或缺，所以不能长时间休息。也就是说，要等他们累得不行了才放假。在暑期（或者更长时间）里同另一家单位互换系统管理员能有助于奖励有价值的雇员，缓解连续一线服务造成的压力。交换过来的系统管理员可能要花点儿时间才能跟得上您这边基础设施发展的速度，但是他们可以从自己单位的环境里带来点子和做法，改进您这里的系统管理工作。

如果是一次跨国交换，那么可能有必要做做表面资助的样子，避开办正式工作签证的繁琐手续。我们就做过几次交换，交换过来的系统管理员还是由他们原来的单位支付工资。有一次，他们换了住房、汽车、宠物、花草等——除了女朋友之外的一切！

30.3.3 分配和跟踪任务

监管项目最关键的方面之一就是保证任务明确到个人。

要避免两种风险：

- 工作漏球，因为每个人都认为这个工作应该是别人做；
- 重复劳动造成资源浪费，多个人或者小组没有协调就在同样的问题上做工作。

工作可以一起来做，但以我们的经验，责任可不容易分清，每项任务应该明确到个人。这个人不必是主管或者经理——只要是个愿意做协调员或者项目经理的人就行。

这种方法有一个重要的作用，就是它含蓄地明确了谁实现了什么，或者谁做了哪些修改。如果您想要搞清楚某件事为什么是以某种方式做的，或者为什么某样东西突然不一样了，或者突然再也不干活了，那么这种透明性就变得重要起来了。

要为一项任务“负责”未必意味着如果出现问题，就要当“替罪羊”。如果您的单位做出了这样

www.TopSage.com

的规定，那么您会发现愿意认领项目的人数迅速减少。认领的目的只是为了消除在“谁应该解决这个问题或者情况”上发生混乱。

从客户的角度来看，良好的分工体系，就是能把问题落实到有能力快速完整解决该问题的个人头上。但从经理的角度来看，任务需要偶尔有些挑战性，超过承担者的能力，这样一来，员工们才能在工作的过程中继续学习和提高。您的工作就是要在发挥员工的力量和对员工有些挑战之间搞好平衡，同时又让客户和员工都满意。

对于日常的系统管理任务，分配任务的关键是一个故障工单系统。这样的系统能够保证故障报告不会无人问津，而是会很快送到最擅长解决它们的人那里。故障工单系统自身就是一个专题，参见 30.9 节了解更详细的讨论。

较大的任务可以成为或者包括完整的软件工程项目。这些任务可能要求使用正规的项目管理工具和业界的软件工程原则。我们这里不介绍这些工具，但是它们很重要，不应该忽视。

有时候系统管理员知道有一项特殊任务需要完成，但是他们没有去做，因为这个任务麻烦。一个雇员如果指出有一项被忽视、未分配或者不受欢迎的任务，那么就可能会被分配去做这个任务。这样的情况造成了一种利益冲突，因为它会促使雇员对这样的情况保持沉默。不要让这种情况在您单位里发生！

30.3.4 管理上级

为了有效地履行您的管理职责（特别是在“领导”领域内的那些职责），您需要得到您的上级的尊重和支持。您要能规定您所领导的小组的结构和人员，包括人事任免的决定权。您需要控制任务分工，包括有权决定什么时候算达到目标，可以重新给员工分配任务。最后，无论在上级单位还是整个世界面前，您都要负起代表小组的责任。

上级经常对于系统管理工作的内容没有概念。使用您的故障工单系统让他们获得这方面的信息，它能在您的上司争取更多的人手或者设备时帮上忙。

即使在没有特别目标的情况下，保留良好的工作记录仍不失为明智之举。经理们（特别是没有技术背景的经理）经常对完成一项任务的难度或者所要的时间估计不准。对于故障排除任务来说，这种不准确性尤为突出。

尝试把期望定得切合实际一些。如果您对计划自己的工作没有太多经验，那么对大型或重要任务则需将估计时间延长至原估计时间的 2～3 倍。如果一次升级工作在两天而不是三天里就完成了，那么大多数用户会感谢您，而不是您估计能一天做完结果两天才完的时候抱怨您。

软件许可证条款经常与一家公司对这个正版软件的实际用法不符，上级倾向于装作看不见。参见 30.12.7 节了解如何在政策和实际之间处理这类差异的建议。

对于棘手的安全政策，总是难以从管理层那里获得支持。加强安全无疑将会给用户带来许多不便，而用户无论是在人数上还是在抱怨能力上都远远胜过系统管理员。安全性增强可能导致用户的工效下降，在实现一种您所提议的安全调整之前，要做一次风险分析，确保管理层和用户都能理解您为什么建议这样做。

影响用户的任何安全调整（例如，远程登录从用口令转为用 RSA/DSA 密钥）都要预先通知并做好文档，同时在转换期间必须得到很好的支持。文档应该易于理解，并为新系统提供详细的处理方法。当您首次接到新系统上的时候，需要考虑额外的工时，好让您能受理那些没有读过通知邮件而感到恐慌的用户。

30.3.5 化解矛盾

经理所负责的几项事务一般就是在遇到困难的情况下与人（通常是客户或者员工）搞好关系。我

www.TopSage.com

们先看看一般的方法，然后谈谈对待“难缠”客户（有时候叫做“牛仔”）的特殊情况。

系统管理工作领域的矛盾经常发生在系统管理员和他们的客户、同事或者供货商之间。例如，一个客户对提供给他们的服务不满意，一个商户没有按时交付所承诺的材料，一位同事没有按您期望的去做，或者一个工程部门坚持认为自己需要掌握其台式机上安装的操作系统的配置。

一、调解

大多数人都不喜欢讨论矛盾，甚至不承认存在矛盾。当怒火爆发的时候，一般都是因为矛盾解决得太晚了，不满的情绪已经持续了很长一段时间。在这个积攒不满的阶段，发生矛盾的双方有机会形成一种因愤恨引起的健康负担，而且会揣测对方的不良动机。

有调解人参加的面对面的会议有时候能够化解这种情形。把开会内容限制在单一主题上，并且限定会议时间为半个小时。这些措施降低了会议蜕变成一种无休止牢骚会的机会。

一旦问题已经明确，而每一方的愿望都已经表达，那么就要努力达成一致，以各种方式去防止这个问题再度发生。在实施这个解决方案之后，要复查矛盾发生的情况，确保解决方案有效，且能长期贯彻执行。

二、难缠的用户和部门

在把受到严格管理的系统引进的过程中，经常会导致矛盾的发生。偏技术的用户（有时候是整个部门）可能感觉到集中式的系统管理不足以满足他们对配置的需求，或者他们对自己使用的计算机自主控制的需求。

您首先要推动的可能就是要迫使难缠的用户接受标准的配置，从而把支持他们所花的代价和时间降至最低。不过，这样的一种铁腕方法通常最终让用户和系统管理员都不高兴。要记住，难缠用户的要求经常完全合法，而且系统管理员的工作就是支持他们，或者至少不会让他们的日子更困难。

对这个问题最称心的解决方案就是找出难缠用户不愿接受被管系统的深层原因。在许多情况下，您可以解决他们的需求，让难缠的人回头。

另一种整合策略是自力更生。允许难缠的用户或者用户组为所欲为，同时要他们都清楚地明白，他们也必须为保证定制系统的运行而负责。安装一个防火墙来保护您所控制的系统不会被可能源自于难缠人士的网络的入侵或者病毒所损害。不要收拾难缠用户的烂摊子。

要让在难缠用户网络中的所有人都签署一份书面的政策文件，其中制定了若干安全方针。例如，如果他们的系统影响了单位里其他的地方，那么他们的网络就会被断开，直到他们打过补丁，不再影响生产网络为止。

在处理工作中越来越多的笔记本电脑时，需要有创造性的系统管理。您必须找到给这些（不可信的）设备提供服务的多种方法，又不会威胁到系统的完整性。用一个单独的网络可能是个不错想法。

所有的站点都有自己的“技术前沿型”用户，他们沉溺于立即获得最新的东西所带给他们的满足感。这样的用户愿意去面对 beta 版软件的不便以及预发布版软件的不稳定，只要他们的软件是最新的就行。找出多种方法来解决这些人的问题，要把他们当作有用的资源而不是您身边的刺儿头。他们都是测试新软件的理想候选人，而且经常愿意把 bug 报告反馈给您，让问题得以解决。

30.4 行政的职能

在这里的“行政”工作包括没有直接给单位的产出做贡献的所有任务，比如会计、人力资源、秘书、销售、采购和其他后勤工作。

根据您所在单位的不同，行政上的任务可能全部由您负责，也可能全部由您的上级单位执行，或者介于上述两种情况之间的某个位置。但是必须以某种方式完成这些任务。

在自我管理的小组里，这些任务会分散到各个组成员的头上。不过，对于非常熟练且很难找的

www.TopSage.com

IT 专家来说，这可能不是一种高效的用人方法。此外，许多人还讨厌行政工作。法律的要求和单位的要求都施加了很严格的规章制度，而有创造性的专业人员不一定对这些制度感兴趣。作为老板有一项重要的工作，那就是保证行政工作要明升暗降、由专人负责、外包或者至少公正地在系统管理员中间分担。

我们这里只介绍几种常见的行政工作，因为大多数行政工作是由上级指挥的，没有商量的余地。

30.4.1 销售

如果您考虑为一个新的或者已有的客户做一个项目，需要物色一个相当于技术销售的人，他能理解潜在客户的需求、懂得您的团队能提供什么样的服务和产品，而且知道团队的能力（忙闲）。没有什么比承诺做不到的事情更糟糕的了。如果老板要参加与客户的协商，那么技术人员应该事先让老板掌握详尽的资料。

30.4.2 采购

在许多地方，系统管理团队和采购团队是完全分开的。如果采购团队没有咨询过系统管理人员就做决定的话，那就不好了。如果采购决定是由上级主管在高尔夫球场上，或者在他们最喜欢的球队揭幕战赛场，坐在由供货商给他们提供的包厢里做出的，那就更糟糕了。

讲个典型的故事：一家磁带存储公司负责 IT 的副总以为他知道一个服务器大项目里需要什么样的磁盘存储硬件。在星期天的高尔夫球场上，他没有咨询过系统管理人员就和 IBM 签订了一份合同。IBM 磁盘阵列要连接到 Sun 服务器上，IBM 的销售人员信誓旦旦地说盘阵一定没问题。结果，设备驱动程序还是测试级的代码，几乎不能干活。在接下来的 3 个月里，IBM 忙着改驱动程序，而该公司的重要服务器宕机的时间比开机的时间还要长。重大项目被延期，在公司里其他部门的人看来，IT 部门就像个傻瓜。存储上犯的大错也影响到了公司的根基——他们在自己的领域内所占市场份额从最大的几家公司之一跌落到了最小的公司之一，原因就是他们的主打产品在市场上落后了。

对于在本单位环境里的兼容性，各个商家的优势（特别是第三方的转销商），还有某种设备的可靠性方面，系统管理员都能提供很好的信息。在 PC 市场上，可靠性信息特别关键，因为在这个市场里的价格都颇具竞争力，而质量往往被完全忽视了。

系统管理员需要知道正在订购的都有哪些新硬件，这样一来，他们就能确定如何把它集成到目前的基础设施中，并能预料到支持它需要什么样的项目和资源。

注意，决定一个系统是否能得到支持并不是系统管理员的工作。您可以向您的客户推荐，但如果他们需要一种特殊的设备来达到自己的目标，那么如何能让设备发挥作用就是您的任务了。您或许不得不多雇人或者怠慢其他系统。给您的老板提供多种选择，让上级去决定他们更愿意在什么上面做出妥协。

对默认情况下必须购买最低报价设备的单位（例如，政府机构和一些州立大学）来说，系统管理员的参与尤其具有价值。大多数采购系统能让您指定评估标准。应该确保包含诸如“必须与当前环境兼容”或“必须可以运行 XYZ 软件”这样的排除条款。

加一个硬件（有时候是加一个软件）所多产生的成本和影响是不固定的。它是该体系结构中的第 60 个产品还是第一个产品？它有足够大的本地磁盘用于存放系统文件吗？它有足够大的内存用于运行现在臃肿的应用软件吗？有空闲的网络接口插上它吗？它使用全新的操作系统吗？

类似这样的问题往往在强调一个更为基本的问题：您是墨守成规，只从您当前的厂商那里购买设备，还是尝试从一家新厂商（他们可能会震撼世界，也有可能在一年内就破产）那里购买最新潮的玩意儿？您的单位性质是这个问题的答案。答案不是简单的是或者不是：您必须经常在最新最好的设备

www.TopSage.com

和您满意和了解的机器之间做出复杂的权衡考虑，应该以有条不紊、不偏不倚的方式来对待它们。

如果允许您与厂商进行（正式或者非正式的）协商，作为系统管理员的您往往能比采购部门更出色地完成任务。您可以举出其他厂商的可比产品的价格，或者夸大来年的采购计划，而不应该对此感到害羞。别忘了，销售人员已经抬高了他们的产品的价值。

在一个季度末或者在一家厂商刚调整其产品线之后，可以让您获得相当大的折扣。销售人员经常需要提高该季度的销售量，或者降低快过时的老型号的库存。

30.4.3 账务

让您的财务情况井然有序是另一项重要的行政工作。不过，这项任务和具体单位太有关系了，我们无法详细多说。您要知道法律的要求和单位的要求，并且及时更新您的账目。审计是件很痛苦的事情。

30.4.4 人事

我们已经介绍过聘用和解聘的问题（参见 30.3.2 节的内容），但是要让一位新雇员融入您的单位里则远不止是写一封录用通知函那么简单。您必须了解并且尊重您的单位在职位、试用期、复审等方面的具体规定。另外一些事务还包括给新来的人安排办公桌、电脑、钥匙、账号、**sudo** 访问权限等。

或许更重要的是，当一位雇员提出离职的时候，您必须把上面的过程倒过来再走一遍。在美国，一般辞职要在两周前提出。有些地方（例如，做安全的公司）用不着两周时间，直接可以走人，随即就收回其物理或者网络的访问权限。这可能是快速做内部检查，核实辞职人员没有在您的服务器上留下任何后门的好时候。有些商业软件包（例如 Novell 的目录产品）的卖点就是能够在一瞬间删除前雇员的信息。

当一家公司计划做大规模裁员时，通常不会和系统管理员商量。裁员的那天到了的时候，管理层会喊着让把以前雇员的机器做备份，收回他们的访问权限等。因为雇员心情不好和计划不周会造成硬件丢失。

30.4.5 市场营销

系统管理是一件有意思的工作。如果您工作做得好，那么用户会想当然地觉得计算环境就那么好，没有人会注意到您的付出。但是现如今的世界充满了病毒、垃圾邮件、臃肿的应用软件，又对 Internet 有全面的依赖，所以 IT 人员成了单位里不可缺少的人。

对您满意的客户就是最好的推销手段。不过，还有几种方法，可以在您的单位里以及在更大的范围里得到关注。根据我们自我宣传的经验，我们觉得下面几种方法尤其有效。

- 召开会议，用户可以在会上说出自己所关注的事情，提出有关计算基础设施的问题。您可以通过分析用户所持诉求来做准备，而且在会议开始的时候就您认为最麻烦的话题做一个简短的报告。要提供点心保证良好的到场率。
- 给提问留出充足的时间，保证有丰富经验的工作人员在场回答这些问题。但对于出乎意料的问题不要靠蒙混过关。如果您没准备，不知道答案，最好承认，以后再问个究竟。
- 安排一系列研讨会，由您在系统管理界的同行，或者您单位的终端用户主持。会议每两三个月安排一次，提前公布要讲的主题[2]。

[2] 您可以邀请系统管理界的同行在系列研讨会上做嘉宾报告。要确保他们的报告展示出和您单位里的报告人同样的高质量。

www.TopSage.com

- 参加系统管理方面的大会，通过大会发言或者撰写论文介绍您所开发的工具。这样的报告不仅能获得同行的反馈，而且他们还会向您的客户（以及您的老板）显示出您的工作做得好。

尽管我们已经在前面列出了这些推销技术，但整个团队显然也必须投身于市场推广活动中。

系统管理最终要和人以及人的需求打交道。人际关系此时和在任何生意场里一样重要。同您的客户和同事沟通，并给个人讨论和交流留出时间。如果您支持多位客户，要考虑给每位客户派一个专职人员作客户经理。这种联络应该有责任让客户在总体上满意，而且应该定期与客户的终端用户交流。通过联络，就能得到客户计算环境上发生变化的消息和信息，从而创造了更多接触的机会。

30.4.6 行政事务

行政小组还要做许多后勤方面的小事情。

- 保留固定资产登记。
- 添置家具和其他办公设备。
- 维护咖啡机，组织提供咖啡、糖和牛奶。买咖啡应该用组内的预算来支付，保证每名员工都喝上咖啡好好干。有些公司还有小吃和饮料。
- 给花草植物浇水。
- 和建筑物的物业部门（或者外面的合同工）联系，布置清洁工作、维护 HVAC 等。
- 管理钥匙或者其他进门的工具。

30.5 开发的职能

大站点需要一个健壮、可扩展的计算基础设施，所以系统管理员不会把他们所有的时间都花在一个灵活性差、设计糟糕的劳动密集型系统上。用实际工作来说，这意味着您需要定制的软件、总体结构和正式负责设计和维护它们的人员。

随着您的站点扩大，累积的计算机数量增多，您不是越来越发愁，就是开始想办法让系统管理过程更优化和更有系统性。最常见的（也是最有力的）优化工具就是工作自动化。它降低了手工劳动强度，又保证了任务能始终如一地执行。

遗憾的是，这样的自动化经常以无组织的方式出现，站点慢慢积累了大量随意拼凑的补丁和脚本片段，造成了令人遗憾的结果。这样的软件没有文档，维护它们最终成为了噩梦。

在开始以全局规划的角度来考虑才算有意义之前，必须要有多少台机器，对于这个数字始终存有争议。我们感觉这其实不是个人头数的问题，更多的是一个职业化和工作满意度的问题。老是到处救火、紧紧张张的生活没有意思，会不可避免地同恶劣的服务质量拉上关系。用户的要求变化范围很大，适合于支持金融交易的需求就不适合大学里的学生实验室，反之亦然。您的解决方案必须符合用户的需要。

“基础设施体系结构”这个别扭的词有时候用来形容系统管理工作的高级概念，与日常事务形成对照。这个词的“基础设施”部分是指 IT 已经成为一种用品或者工具。就像电话系统或者水龙头一样，它就应该能干活。如果基础设施坏了，应该很容易找到坏了的地方，并且更换损坏的部件。这个词的“体系结构”部分表明设计这个基础设施是为了特定的目的，它经过裁剪符合客户的需要。

本节我们列出基础设施体系结构的一些特点。要深入了解这个专题，可查阅本章末尾的参考资料。本书第 11 章里可以找到在实现上的一些有帮助的指导。

www.TopSage.com

30.5.1 系统构架的原则

接下来的几个小节介绍了您在规划自己站点的 IT 设计时，应该考虑的一些体系结构上的“最佳实践”。当您要支持的配置很新或者不同寻常的时候，这些原则尤其重要，因为这些情况难以同现实中的同类配置相比较。设计良好的过程要遵守这些原则。

一、过程可再现

系统管理不是一种行为艺术。无论做的是什么，都应该能重复完成，且前后一致。通常这意味着，最底层的变化应该是脚本或者配置程序来做，而不是系统管理员自己做。配置上的变化应该反映在系统管理软件的配置文件里。

例如，设置一台新机器的脚本不应该问 IP 地址和要安装的软件，相反，它应该检查系统配置目录来判定要做什么。它可以输出这些信息要求确认，但要预先定好可供挑选的选择。用户干预得越少，发生人为错误的机会就越小。

但有一点要说清：我们并不是说一个站点里高级系统管理人员所决定的政策要由没有思想的傀儡去执行。可再现性的要求就仿佛您是单位里唯一的系统管理员那样。一般不应该随便决定配置，不留可供审计的记录。如果需要改动什么，则要改核心的配置信息，然后从那向外扩散。

二、留下记录

谁做过什么，为什么目的？如果您的系统有问题，当您可以回到上次能工作的状态，或者至少能确定从那以后改过什么，那么修起来会快得多。除了知道“改过什么”，知道“谁改的”和“为什么改的”也很重要。和那个修改过系统，造成问题的人谈一下，往往能有对问题重要的见解。您可能可以很快退回所做的改动，但是有时候改动的理由很对，退回去只会让情况更糟。

版本控制系统提供了一种跟踪改动的有用方法，我们在 11.5 节详细讨论过。这类系统既有实际数据随时间推移的历史记录，也有是哪个系统管理员做的修改的信息。如果使用正确，每次修改都有一条注释，说明修改背后的理由。自动化的工具能把他们修改的配置文件检入版本控制系统，并且在注释里说明自己的身份。这样一来，很容易就能找出功能有误的脚本，撤销它所做的修改。

另一种有用的做法是某种形式的电子邮件日记。这要么是一条永久性的消息记录，发给您站点上的一个实际的系统管理邮递列表，要么是一个只保存记录日记更新的邮箱。日记提供了一个广为人知的记录配置变化信息的地方，通过电子邮件实现则意味着更新不会给系统管理员增加负担。电子邮件日记的缺点是信息除了按时间顺序之外没有什么结构。

三、认可文档的重要性

实际上，文档对于一个可扩展的基础设施如此重要，以至于我们专门有一个 30.8 节介绍它。

四、定制和编码

使用现成的工具是一种优点，您应该尽可能地使用它们。但世界上没有哪个站点和您的站点完全一样，所以您的单位肯定有一些独一无二的要求。精确满足单位需求的 IT 基础设施需要有一种优势，能提高每个人的工效。

Linux 具有出色的脚本化能力和丰富的开放源代码工具，它是良好基础设施的理想基础。以我们的观点来看，不具备软件开发能力的系统管理员小组是瘸腿的。

五、保持系统干净整洁

系统管理工作不只是安装、添加和配置，也要知道保留什么、抛弃什么以及翻新什么。我们把这个概念称为“可持续化管理”。能在 5 分钟内给您的环境增加一台计算机非常好，而能在 10 秒时间里创建一个新用户账号也不错。但是如果您考虑到将来，能够以一种有组织的方式找到和删除老账号、找到和撤掉老计算机，也是同样重要的。可持续化系统管理意味着，您具有以一种有组织的方式长期做运行工作所需的工具和概念。

www.TopSage.com

30.5.2 管理系统的组成

您的管理系统应该包括以下主要组成部分。

- **自动设置新机器**。这不仅是操作系统的安装，它还包括让一台机器开始投入工作所需的额外软件和本地配置。您的站点不可避免地需要支持一种以上的配置，所以您应该在一开始的规划里就要包括多种机型。本书第 11 章讨论了几种能够帮助进行设置的系统。
- **对现有机器进行系统级的打补丁和更新操作**。当您确定您的设置有问题的时候，需要一种标准化和方便的方法把更新部署到所有有关的机器上。注意，因为计算机不是所有时间都开的（即使是应该一直开机的计算机），您的更新机制必须正确地处理开始更新时没有上线的那些机器。您可以在系统引导的时候检查是否有更新，或者定制做更新。参考 11.6.5 节了解更多信息。
- **监视系统**。您的用户不应该只能靠打电话告诉您服务器宕机了。这样做不仅不专业，而且您也不知道系统已经宕机多久了。第一个打电话给您的人不一定是第一个遇到问题的人。您需要某种监视系统在一出现问题的时候就发出报警。但是报警要有些技巧性。如果报警太多，那么系统管理员就开始无视它们；如果太少，重要的问题会没注意到。
- **通信系统**。不要忘记同用户的需求保持接触，支持他们是您的终极目标——按系统管理员的要求做一切工作。必须要有一个跟踪需求的系统（30.9 节）。用户可以在一个中央的位置找到系统状态和联络信息（一般是在 Web 上），这也会有很大帮助。

30.5.3 系统管理员的工具箱

好的系统管理员是懒（聪明）的，他们通常会尝试让自己的工作自动完成。一位成功的系统管理员会编写脚本，填补从厂商那里获得的工具与网络上下载的工具之间的空白，然后让这些脚本在他们需要的地方都能得到。

大多数地方都有他们自己的 **adduser** 脚本，或许还有一个 **rmuser** 脚本。如果您是一位独自工作的系统管理员，那么您或许要把这些小宝贝放在您自己的**~/bin** 目录下，不断地使用。如果您是一个两人以上团队的成员，那么情况就要复杂一些。采取把工具多复制一份，或者修改搜索路径指向另一个人的**~/bin** 目录这样的权宜之计能在短时间内帮上忙。但您最终需要找到某种持久性的解决办法。

我们对这个问题的解决方案是“系统管理工具箱”。这个工具箱里有所有本地开发的系统管理软件，并且有自己的专门目录树（例如，**/usr/satools**）。最重要的是，我们有一份政策文档，说明了如何编写工具箱的工具。系统管理员之一“拥有”这个工具箱，对政策文档之外的事情做出决定。

当创建这样的政策时，重要的是要保证政策的每一项内容都有其存在的清楚理由。只有人们能理解规则的动机，他们才能遵守政策的精神，而不只是规则表面的内容。

下面列出一些政策条款的例子。

- 公开工具放在**/usr/satools/bin** 下。配置信息保存在**/usr/satools/etc** 里。工具在运行时生成的日志文件和任何别的数据文件都在**/usr/satools/var** 下。静态数据文件在**/usr/satools/share** 里。理由：标准化。
- 工具应该用 Perl 或者 Python 编写。理由：标准化，便于在系统管理员间交接和代码重用。
- 每个工具都必须有自己的手册页。理由：如果一个工具没文档，那就只有它的作者知道怎么用。
- 开发工具时，先写文档，在编写代码前要让管工具的人先审查。理由：在任何工作完成前，先找出重复性劳动和不符合政策的问题。

www.TopSage.com

- 每个工具都必须接受一个**--no-action** 选项，让它只是显示它要做什么，但不实际做任何事情。如果没有实现这样一个选项，那么工具就必须至少在出错时提供一条适当的消息。理由：有助于防止灾难发生，特别是像 **rmuser** 或者 **mvhomedir** 这样可能造成的灾难的工具。
- 所有的工具都必须在 **svn://server/satools**（一个 Subversion 源代码库，参见 11.5.5 节）的版本控制之下。理由：可以多人同时在工具上开展工作，清楚的版本控制简化了源代码管理和 bug 处理的工作。

现实中的政策文档可能会有几页纸那么长。

您可能想写一个小小的工具箱管理工具，把新工具从 Subversion 库中安装到**/usr/satools** 目录树里。为了控制质量，这个管理工具也可以尝试做一些检查是否符合政策的工作。例如，它能核实每个工具是否安装了文档。

30.5.4 软件工程的原则

因为系统管理脚本不一定像个“真的”软件，您可能倾向于认为它们不需要达到典型的软件工程项目在筹划和形式上的程度。这也对也不对。在达到一定的复杂性和重要性之后，您确实需要严谨地对待这些项目。

Frederick P. Brooks, Jr.所写的软件工程方面的经典《人月神话》（本书注释版已由人民邮电出版社出版）是一本易于阅读的好参考书。书中呈现给读者一个有趣的事实，作者讨论的从 20 世纪六七十年代就有的软件工程问题至今仍然存在。下面是软件工程原则和失误的复查表（没有特别的顺序）。

- 应该清楚地界定项目的范围，并规定好完成项目的时间计划。
- 责任应该明确规定。项目上工作的系统管理员、系统管理经理以及最终的客户都必须就项目的目标、范围和可能的风险取得统一。
- 在开始编码之前应该写好预备文档。
- 软件各个组件间的接口应该事先规定好且有文档。
- 重要代码应该先有原型实现并经过测试。在开发的尽可能早的阶段就能得到软件用户的反馈非常重要。
- 状态信息应该如实交流。没人喜欢意外，坏消息最后一定会出现。
- 每个项目都包含事后分析。从错误和成功中学习提高。

软件项目的最优管理一直是个热闹的话题。在过去 10 年中，强调快速原型实现、及早开发、增量功能扩展和消除开销的“敏捷”方法特别受开发人员重视。其价值远不止人们普遍同意的那样，但是许多思想似乎像是常识，这种轻量级的方法更适合于系统管理领域里出现的较小的项目。

30.6 运行的职能

运行部门负责 IT 基础设施的安装和维护。运行部门和支持部门一起组成了 30.2 节所说的“执行”部门。根据规定，运行部与计算机和线路打交道，而支持部与人打交道。

运行只包括现有的服务，所有“新东西”应该先由开发部设计、实现和测试，然后再部署到一个生产环境里。

运行工作集中在为客户创造一个稳定、可靠的环境上。可用性和可靠性是运行工作的关注重点。运行人员不应该做实验或者在星期五下午搞维修或者改造。那样的话，发生故障（以及在周末除了客户之外没人注意到问题）的机会可就大多了。

www.TopSage.com

30.6.1 瞄准最少停机时间

许多人都依赖我们所提供的计算基础设施。大学里的一个系或许能允许一段时间没有自己的 Web 网站，而一家像 Amazon.com 这样的 Internet 邮购公司则绝对不行。有些人不会注意到您的打印机服务器宕机了，但要是一个急着赶在最后期限前提交研究论文或者提案的学生或者教授的话，就会非常不满。即使是在大学的环境下，不能访问电子邮件也通常会让每个人发怒。中央文件服务器则是另一个可能的灾难发源地[3]。

在有些地方，您需要提供应急服务。在一个商业环境中，这种服务可能意味着每周 7 天、每天 24 小时都有富有经验的系统管理员在场。但即便是像大学里的系这样一个非商业性的环境，在一个漫长的周末没有对主系统做故障切换也不是好事。

即使您没有经费做到 24/7 的覆盖范围，也应该做好准备，充分发挥碰巧在夜间或者周末在场的任何系统管理员的作用。轮流使用的呼机或者其他通知系统经常能起到相当大的应急效果。要保证用户都知道且能方便地用上这样的应急措施，例如，用电子邮件别名 support-pager 把消息转给轮流使用的呼机。

30.6.2 依赖文档

为了正确地报告可用性或者持续运行时间，您一定要知道自己的强项和弱点（包括您所部署的硬件的可靠性），而且要知道 IT 系统对其他硬件、软件和人员的依赖性。

- 电源：独立电源和电路、电涌和短路保护、备份电源系统（如发电机和 UPS）、建筑物的电路、特定于设备的电源接线图。
- 网络：建筑物的布线、备份线路、ISP 的客户服务、网络拓扑、单位内有其专门网络管理职能的其他小组的联络信息。
- 硬件：高可用性系统及其使用规程、热/冷后备、空备件、硬件维护合同。

30.6.3 改变老硬件的用途或者淘汰老硬件

为了维护您的基础设施，您必须购买新机器、将旧机器用于别的用途，并且淘汰老机器。我们在 30.4 节介绍行政工作的时候讲过采购，但淘汰老机器似乎更像是运行部门的事情。

每次提高硬件性能时，软件通常都会变得更大和更复杂，把硬件增加的性能抵消掉。例如，Windows 95 和 Windows 98 只要 32MB 的内存就能运行了。Windows 2000 没有 128MB 则跑得很差，现在运行 Windows XP 的笔记本跑个任务管理器似乎都要 256MB 以上的内存了。当然，这种臃肿问题并不限于 Windows，目前的 Linux 桌面环境也需要大量的内存。

因为用户和管理人员经常会犹豫是否升级过时的设备，您有时候不得不要主动提议升级。财务信息是最有说服力的证据。如果您能够书面证实，维护老设备的成本超过了更换它们的成本，您就能从思想上破除人们对升级的抵制。有时候，为了节省让所有不同的操作系统和软件版本保持最新所要花费的时间和精力，替换掉不同类型的硬件也是会有帮助的。

廉价的 Intel/PC 硬件已经成为桌面机器的标准体系结构基础，特别是现在苹果公司都开始提供 Intel 硬件的机器了。这些年来，随着 PC 的流行，计算设施的成本已经从硬件转向了软件和支持。

[3] 如果一台文件服务器发生故障的时间超长，不高兴的用户会坚决要求用自己的个人硬盘而非文件服务器来保存数据（至少要等到其本地硬盘发生故障，又没有备份，使得他们为此恼怒的时候）。

www.TopSage.com

大家网 TopSage.com

您可以通过保留老系统的方法，让老系统用户的过渡变得方便。让老系统还有电，但是系统管理团队提供的支持越来越少。停止维护老机器上的硬件，让它们自行退出历史舞台。

您可以把比较老的机器用于专门的服务，这些服务不要求太多的硬件资源（例如，打印、DHCP或者许可证服务器）。只要保证您拟定有计划，用于维修或者更换这些机器，或者在发生问题时把服务转移到另一台服务器。

大学经常会从想得到减税的商业公司那里获得捐赠的老计算机。对于这种情况，正确的回答往往是“不，谢谢，我们不需要2000卷九道磁带以及放置它们的磁带架”。几年前，布达佩斯的某个大学得到了一台捐助的IBM主机。他们没有拒绝捐助去买更快的PC，结果他们把头一年的预算都花在了运输和电气布线上面，随后几年的预算则都是维护费。这简直就是得不偿失。另一方面，许多大学与当地计算机行业界建立起了紧密的关系，并且得到了许多贵重的硬件捐赠——这些硬件只是去年的型号，并不过时。由此可见，考虑获得捐赠后总体的成本，就掌握了说“不，谢谢”的时机。

如果您是一家公司，而且有富余的计算机设备，请考虑把它捐助给当地的学校或者学院。他们经常没有预算经费来让自己的实验室跟上潮流，他们会欢迎您提供的老PC的。

不能提供给慈善机构的淘汰计算机必须正确地回收，它们包含有毒材料（例如，铅），不能扔到普通的垃圾箱。如果您的单位没有一个设备部门处理旧设备，那么就不得不把这些设备送到当地的回收中心做报废处理。美国的废品处理价格大约是每个系统10美元。

30.7 支持工作

支持部门的任务是和人打交道，这些人使用并依靠您所维护的计算机系统。除了提供通常的问询或者热线支持之外，这个部门也可以提供咨询服务或者做专题培训。在理想情况下，这些辅助性的服务提高了客户的自立能力，减少了寻求支持的次数。

30.7.1 可用性

良好的IT支持意味着有合格的人员在用户需要他们的时候提供帮助。

大多数问题都不大，排队服务没问题。其他阻碍工作的问题必须立即得到关注。从需求跟踪系统返回的自动回复，以及从录音电话中传出正常办公时间的说明，都会造成用户的烦恼。要保证用户如果有需要，总能有一条最后求助的途径。下班之后，轮流在系统管理员中使用值班手机通常就足够了。

遗憾的是，出色的支持培养了用户的依赖性。用户很容易养成一有不对劲儿就咨询问讯处的习惯。如果您认识到，有人使用支持系统获得的答案，他们从手册页或者Google很容易就能获得的话，就开始以援引相关手册页或者URL的方式来回答他们的问题。这个做法对学生很有效，对经理们的效果可能差一些。

30.7.2 服务范围

由管理支持组提供的服务必须进行明确规定，否则用户的希望将无法与现实相符。下面是一些要考虑的问题：

- 响应时间；
- 在周末和下班时间提供服务；
- 家庭电话（支持家里的机器）；
- 奇怪的硬件（一些不常见的种类）；

www.TopSage.com

- 陈旧的硬件；
- 所支持的操作系统；
- 标准配置；
- 备份磁带的作废；
- 特殊用途的软件；
- 勤杂人员的日常杂务（清洁屏幕和键盘）。

在Linux的世界里，每个用户都能获得源代码，并且能为自己的机器轻松地构造一个自定义的内核。您必须在给您单位的策略里考虑到这一点，或许应该有几种专门的内核配置作为标准。否则，要达到随单位规模的增加保持易于维护和扩展的目标，就会遇到一些严峻的障碍。鼓励有创造性和精通操作系统的雇员就其工作的需求提出内核的修改意见。让他们成为标准化过程中的一个环节，您就不必为每位工程师分别维护不同的内核了。

用户除了了解提供什么服务外，还必须了解用于管理多项工作先后的优先级方案。优先级方案通常都留有一定的调整空间，但在设计该方案时应尽可能地考虑周全，没有或者很少有例外。下面是一些和优先级相关的变量：

- 客户付费购买/合同签订的服务等级；
- 受影响的用户数量；
- 受影响的用户的重要性（这一点很微妙，但是一般来说您不想惹恼任何付钱给您的人）；
- 受影响的用户的反响程度（唧唧歪歪）；
- 最后期限的重要性（晚上的家庭作业与研究资助提案或者合同之间的对比）。

虽然所有这些因素都会影响到您的服务评级，但我们推荐一组简单原则和一些处理特例的常识。基本而言，我们做下面的优先考虑：

- 许多人不能工作；
- 一个人不能工作；
- 要求改进。

如果两个以上的请求都有最高优先级，又不能平行解决，那我们就根据问题的严重性来决定首先解决哪个问题（例如，电子邮件失灵的话会让几乎所有人都不爽，而有些Web服务临时不能访问则可能只影响到个别人）。排入更低优先级队列中的问题通常按照先入先出的方式来处理。

您可能想要衡量做支持工作的人员的常识水平。我们通常问他们这样的问题，“如果大老板走进办公室，你正在处理一个紧急问题，他要你修理打印机，因为他需要打印几张要在会上讲的幻灯片，去机场的出租车将在15分钟内离开，当然，除了你之外，IT组里的每个成员都出去吃午饭了。这时候你该怎么办？”任何一个做支持工作的理性的人都会（至少也会在重新考虑过后）认识到，同一个已经处在繁多压力下的人去争论什么没有什么意义。为了稳妥起见，合理的做法是询问一下，不去管受损的中央服务器是否可以。如果回答是可以，那么就放下手头所有的事儿，把幻灯片打印出来。

我们接着问，“当大老板几个星期之后又让你碰到了类似的情形，你会怎么办？”同样，仍然没必要争，但是这次在他转而讨论他需要的支持之后，安排一次和他的会面，这样做可能是个好办法。或许他确实需要派过一个人去支持他个人，而且如果他愿意对此付费，这又没什么坏处。在他正担心赶不上飞机的时候，不要尝试去这样讨论。

用户一般会认为他们所有的重要数据都储存在备份磁带上，并且被永久保存。但是备份磁带并不能无限维持下去，磁介质的寿命有限，之后就变得难以读出数据（您必须定期重写数据，如果您想要长期保存的话，可能还要写到新介质上）。备份磁带也可以被索取，您的机构可能不想永远都留着老数据。最好和负责做决定的人一起起草一份书面的协议，指出备份必须保留多长的时间，是否要做多个拷贝（要求？可以？不用？），那些拷贝是否必须保存在不同的地方。

您应该让系统的用户掌握这些信息，因为它提高了备份的理想寿命。还要提醒用户注意，如果他

www.TopSage.com

们觉得需要更好地保护自己的数据，那么就应该自己多加防备。

30.7.3 技能要求

对于一个有经验的用户来说，如果联系到的支持人员一边问“您把电源线插上了吗？”一边在后面疯狂地搜索客户服务数据库，那就没有什么比这更让人恼怒的了。另一方面，让经验最丰富的系统管理员向一个初学的用户解释如何在某种字处理系统中找到删除键，这也是一种资源浪费。

在我们的环境中，每个员工每周都会被派去做若干小时的技术支持工作[4]。支持人员在每周计划中留出这段时间，只要分配了支持请求，就从这里面抽时间。任务要按照解决问题所需的技能，以及每个人每周做支持工作的时间预算剩余量来分配。

为了让这个方案获得成功，要保证技能名单要均衡。从长期来看，支持员工中的每个成员都必须能够完成分配给他们做支持工作的时间。一般来说，在技能名单中出现多次的员工更“有价值”。不过，员工技能不多也没有错，只要您给他们分配了足够的工作去做即可。

一份准确的技能清单有助于您核实有充足的专业技术力量来应对休假和病假。随着问题的出现并由员工里的成员解决，您就可以建立技能清单。清单里要包括任务、员工的名字以及表现出的经验水平。

技能应该以适当的抽象级别来规定，既不要太专，又不能太泛。下面所举的技能清单样表的细致程度很合适：

- 创建用户、删除用户、设置口令、改变配额；
- 创建 CVS 或者 SVN 账号；
- 从备份恢复文件；
- 把新硬件的驱动程序集成到 RIS（Windows）；
- 把 Windows 应用程序以 MSI 格式打包；
- 在 Linux 上创建和安装软件应用包；
- 分析日志文件；
- 调试邮件服务器的问题；
- 调试打印问题；
- 调试一般的硬件问题；
- 设置 DNS 项；
- 管理软件许可证；
- 回答 Windows 安全问题；
- 回答 Linux 安全问题；
- 解决 Samba 相关的请求；
- 配置 DHCP；
- 配置 LDAP 服务器；
- 增删 Web 网站（配置 Apache）。

30.7.4 时间管理

系统管理工作的内容在一天内变换的次数比大多数工作一年里变换的次数还要多。支持用户的人

[4] 在默认情况下，要把每个系统管理员 25%的时间分配去做支持工作。当一个项目要求一名特别的系统管理员全时投入的时候，我们可以打破默认的安排，在项目期间让这个人不做支持工作。

www.TopSage.com

员承担了这些任务中的大多数。每个系统管理员都需要具备良好的管理时间的能力。没有这些能力，您就跟不上每天的任务，于是会变得沮丧和消沉（如果已经沮丧和消沉了，那么会变得更厉害）。

系统管理员最终的倦怠在所难免。大多数系统管理员只能干几年。没有人愿意没完没了地被电话找，又接连不断地被人嚷嚷。找到有效管理自己时间的方法，让您的客户满意是一种双赢的局面。

30.8 建立文档

正如大多数人都接受锻炼和带叶绿色蔬菜有益健康一样，每个人都赏识好的文档，对其重要性也有模糊的认识。遗憾的是，这未必意味着他们可以在没有催促的情况下写文档或者更新文档。

我们为何要那么重视文档？

- 文档降低了发生单点故障的可能性。有工具能立即部署工作站，仅用一条命令就可以把补丁发布出去自然好，但如果没有文档，而写工具的专人又在休假或者已经离职了，那么这些工具几乎就没有用了。
- 文档有助于操作的可重复性。如果做法和步骤没有保存在单位的存档里，它们就不太可能会得到完整一致的遵守。当系统管理员不能找到如何去做的相关资料时，他们只好临时凑合。
- 文档能节省时间。写文档的时候不会觉得节省时间，但花了好几天重新解决一个以前已经解决过，但忘了怎么解决的问题时，大多数系统管理员都会承认这个时间花得值。
- 最后，也最重要的是，文档增强了系统的完整性，让后续的修改都按符合系统工作所应该遵循的方式来进行。只是根据不全面的了解所做的修改，经常无法很好地与系统结构保持一致。随着时间推移，变化越来越多，即使是在系统上工作的系统管理员也会把它看成是乱七八糟的拼凑。最终的结果往往是要推倒一切重新开始。

本地建档工作能够服务于许多用途。您进入机房重新启动某台服务器时，所面对的却是一排排机架上看上去十分类似而实际各不相同，并且没有任何标记的硬件设备吗？或者，在不得不安装某个您以前摆弄过的硬件时，您对此所能记起来的一切就是这个硬件很难配置吗？

本地文档应该放在明确的地点。根据运行规模的不同，这个地方可以是文件服务器上一个安装到所有机器的目录，或者可能是特殊系统用户账号的主目录。

30.8.1 标准化建档

我们的经验表明，维护文档最简单和最有效的方法就是建立简洁、轻量级的标准化文档。不要为您的单位写一本系统管理手册，而是写许多只有一页纸的文档，每份文档的内容涵盖一个专题。从总体图开始，再细化为多个部分，每部分包括更多的信息。如果您必须详细介绍某个部分，那么就多写一份这样的单页文档，专门介绍特别难或者特别复杂的步骤。

这种方法具有下面这几个优点。

- 您的老板可能只对环境的总体设置感兴趣。回答上级的问题或者做一次管理上的探讨，就这些文档也够了。不要提太多细节，否则您会引得老板去干预其中的内容。
- 对客户也一样。
- 单位里的新员工或者接管新任务的人需要对基础设施有个全面性的了解，这样工作效率更高。让这样的人陷入大量细节没什么帮助。
- 使用正确的文档比浏览长篇累牍的文档效率更高。
- 如果只需更新一页内容，那么保持文档最新就更容易实现。

最后一点特别重要。保持文档最新是一项巨大的挑战，时间不够的时候往往首先抛弃的就是更新

www.TopSage.com

文档。我们发现有两个专门的方法能保持文档的更新。

首先，设定期望，要求文档言简意赅、朴实无华。保持切题，最重要的事情就是把信息记录下来。没有比指望把设计原理都写成军事规范论文而让文档更快失效的事情了。太多的文档等于没有文档。

其次，把文档融入到处理中。配置文件里的注释是最好的文档。您需要它们的时候，它们一定在，维护它们实际上根本不花什么时间。大多数标准配置文件都能写注释，即使是那些特别不喜欢注释的配置文件经常也加进了一些注释。例如，**passwd** 文件 GECOS 字段的标准内容（办公室、电话号码等）往往不是您可以加在那里的最有用的信息。随心所欲规定您站点自己的习惯（但是在本例中，要记得用户能够改动这个字段的内容，所以您不一定能相信那里保存的信息）。

本地构造的工具要求有文档作为其标准配置信息的组成部分。例如，设置一台新计算机的工具可以要求有关计算机的属主、位置、支持状态以及收费等信息，即使这些实际情况不直接影响到机器的软件配置。

文档不应该造成信息冗余。例如，如果您维护了站点级的主控配置文件，它列出了机器名称和相应的 IP 地址，那么就不应该有别的地方可以手工更新这些信息。在多个地方做更新不仅浪费时间，而且随着时间推移，肯定会出现不一致的情况。在其他场合和配置文件里需要这些信息的时候，可以写一个脚本，从主控配置中获得（或者更新主控配置）。如果您不能彻底消除冗余，至少要清楚哪个来源是权威的。写工具程序来发现这种不一致性，或许可以从 **cron** 定期执行这样的程序。

30.8.2 硬件标签

有些文档最适合于写在一张纸上，或者标在一个硬件上。例如，一个完整系统或者网络基础设施的应急步骤如果保存在一台死机或者访问不到的机器上，那么就不太会有用。

每台计算机都应该做到不用人打开和登录进去就能辨认出来，因为不一定总有可能这么做。每台工作站有唯一的标记（主机名、IP 地址），在它上面贴个标签，上面有咨询台的联络信息。

在服务器机房，所有的系统及其外设都必须标上主机名（如果那里有不同域的机器，那么要用完整域名）。在机器（特别是紧紧放在机架上的机器）前后都贴上标签，以便很容易找到机器的电源开关。

如果您的环境包括许多不同种类的系统，再增加有关体系结构、引导指令、特殊的按键次序、其他文档链接、厂家热线或者负责人电话号码的信息。记下按键次序显得有点儿傻，但是服务器经常会连到一种过时的终端或者控制台服务器上，而不是一个专门的显示器。

确保在您的中央记录或者资产数据中留有所有这些小粘贴标签内容的副本。如果您通过一条连到控制台服务器（console server）的 TCP/IP 连接来管理您的机器，而不是上班整天呆在一个嘈杂的机房里，那么手边的这些信息会很方便。

和每台机器有关系的其他硬件（硬盘、调制解调器、磁带机等）也要标上主机名。如果几个相同的外部子系统都连接到同一台主机，要做到每个标记都是唯一的[5]。在我们安装的系统上，甚至连 SCSI 线和光纤通道线都做了标记（当然，也包括服务器的各个不同端口），这样一来，我们就真的知道哪个设备连到哪个接口。

如果一台主机很重要（例如，一台主服务器或者一台重要的路由器），那么还要标上其电源断路器的位置。如果系统引导需要一张软盘或者闪存卡，那么也要指出它的位置。主文件服务器应该有硬盘设备名、分区表、安装点和可用备份超级块的信息。把这些信息标在硬盘子系统本身上面，或者保存在一个众所周知的地方，或许就放在机房里。

磁带机应该标上访问它们需要用的设备文件和命令。列出磁带机要求用的磁带类型、购买它们的

[5] 最近，我们的一个同事换错了 RAID 盘阵的写缓存备份电池。幸好这个错误只造成了性能下降。但是如果您换错了 RAID 5 盘阵中的硬盘，而又没有发现，接着硬盘第二次出现了故障，想象一下会怎样。

www.TopSage.com

最近地点，甚至大致的价格也是个好主意。

打印机应该标上它们的名字、简要的打印指令以及它们所依靠的主机。打印机经常带有网络接口，是完整的网络设备，但是它们仍然要依靠一台 Linux 主机来缓冲打印作业和进行配置。

网络连线必须一丝不苟地写清楚文档。给所有的网线加上标签，配线板和墙上插座应有标注，然后网络设备也有记号。始终要让您的布线工程师们能够很容易地更新文档，在配线室的墙上挂上一只铅笔和若干张表格，于是把一根线从一台设备换到另一台设备的情况记下来也不费什么事儿。之后，您应该把这些数据转换成在线的存储资料。

如果您的网络设备（例如，路由器和交换机）能够通过网络重新配置，那么就又多加了一层复杂性。虽然您现在可以从舒服的办公室里把机器在子网间移来移去，但是记录文档也就愈发重要了。此时要更加仔细，因为您可以更快和更彻底地把基础设施中大得多的组成部分搞乱。

30.8.3 用户文档

给新用户准备一份打印好的文档是个好主意。文档的内容应该包括本地习惯、报告问题的规程、打印机的名字和位置、做备份和系统停运的时间等。这类文档可以节省大量的系统管理和用户服务的时间。您还应该把这些信息放到网上。打印出来的文档更可能由新用户来读，而 Web 网页则在出问题的时候更容易去参考。上述两种形式的文档都要有，并且要定期更新它们。没有什么比一份过期的在线文档或者 FAQ 更让人烦恼的了。

除了给本地的计算环境建立文档之外，您可能想准备些有关 Linux 的介绍性材料。在大学环境里这样的材料很重要，因为用户群都是暂时性的，而且经常是 Linux 盲。

我们提供了打印版的单页纸条，内容都一模一样，上面列出了我们的用户群常用的命令和应用。

30.9 需求跟踪和故障工单系统

如果您的团队人数很多，那么你将会发现，用一个需求跟踪系统（request-tracking system）给您的工作编排增加某种正式的结构会有帮助。即使您的 IT 部门只有一个人，这样的一个系统也对跟踪所有未解决的问题和需求都非常有帮助。需求跟踪系统也是一个很好的历史信息来源，能反映您的团队已经完成的任务，他们是一种跟踪您内部的 to-do 列表和项目的方便方法。经理一直都想知道您正在做什么，大多数系统都能提供报告，反映出您的工作量、资源以及对更多人员和经费的合理要求。

这些系统有各种不同的称呼，最常见的是“需求跟踪系统”、“故障工单系统”或者“缺陷跟踪软件”。所有这些称呼都指同一种系统。网站 linas.org/linux/pm.html 给出了各种开放源代码实现及其分类。

30.9.1 故障工单系统共同的功能

故障工单系统通过各种接口（电子邮件、Web 表格和最常用的命令行）接受请求，从提交到解决一直跟踪它们。经理可以给员工小组或单个员工派工单。员工可以查询系统看到在排队等待解决的工单，或许自己能解决其中一些工单。用户能够查到一个请求的处理状态，看到是谁在处理这个工单。经理能够从中获得总体的信息：

- 开出的工单数量；
- 关闭一个工单的平均时间；
- 系统管理工作的工效；
- 未解决（窝工）工单的百分比；

www.TopSage.com

- 按时间算的解决问题工作量的分布情况。

保存在工单系统内的需求历史成为了 IT 基础设施出的问题及其解决方案的历史。如果这一历史信息很容易查出来，它就成了系统管理人员的一种宝贵资源。已经解决的故障消息可以发给管理员新手和培训学员、可以加进一个 FAQ 系统或者只作为日志记录。对于新来的员工来说，能看到已关闭的工单会非常有帮助，因为那些工单不仅有技术信息，而且举例说明了与客户沟通交流的适宜方式。

和所有的文档一样，您的工单系统的历史数据也有可能在法庭上对您的单位不利。按照您的法律部门为文档设定的处置方针来处理工单的历史数据。

大多数需求跟踪系统自动接受新的需求，并给它们分配一个跟踪号，提交者可以用这个号来跟踪或者查询请求的处理状态。自动回复消息应该清楚地说明它只是一个确认。后面应该紧跟一则由真人发来的消息，说明处理这个问题或者请求的方案。

30.9.2 故障工单系统的用户接受程度

能立即接收到从真人发来的回复是客户满意度的一个决定因素，即使真人发来的回复没有包含比自动回复更多的信息。对于大多数问题来说，让提交者知道工单已经被一个真人收到，要比立即解决这个问题远远重要。用户能理解系统管理员会收到许多需求，所以在得到系统管理员的关注之前，他们愿意等上一段公平合理的时间。但是他们不愿意被忽视。

我们的故障报告系统使用称为“support”的电子邮件别名。曾有一段时间，我们收到了大量不是不完整就是无法理解的故障报告的“轰炸”。为此，我们编写了一个脚本，向用户提出明确的问题，比如：

- 问题发生在哪一台主机上？
- 该问题会重复出现吗？
- 立即排除该故障有多重要？

1 小时以后用户就对此产生了激烈的反应，而在一天之内我们就放弃了这个系统。其唯一有价值的地方似乎在于，许多用户实际上还是阅读了脚本所提出的问题，所以才对这个脚本发怒了，而之后他们以自由形式报告故障的质量却提高了。

另一个地方处理这个问题的做法是，发出消息，解释在故障报告中什么样的信息是重要的，并且给出了一份无用报告的例子。在收到无用的报告后，给用户的反馈是表示歉意的答复（“对不起，对于解决……我没有掌握足够的信息”）和一份解释性的消息。用户很快就掌握了。

根据我们的经验，表格只适用于高度标准化的任务，比如请求创建或者删除账号。在这些情况下，您的客户能理解您要做工作就必须要求他们提供信息。有些任务一旦填好表格就能完全自动地去做，比如给用户设置一条休假消息。其他任务，如创建账号可以部分自动完成，但要求授权或者其他员工的处理。

30.9.3 故障工单系统

下面的表 30.1 和表 30.2 总结了几种知名的故障工单系统的特点。表 30.1 给出了开放源代码的系统，而表 30.2 给出了商业系统。

表 30.1 开放源代码的故障工单系统

名 称	输入[a]	语言	后端[b]	URL
Mantis	**WE**	PHP	M	www.mantisbt.org
RT:Request Tracker	**W**	Perl	M	www.bestpractical.com

www.TopSage.com

续表

名 称	输入[a]	语言	后端[b]	URL
Scarab	**W**	Java	M	scarab.tigris.org
Double Choco Latte	**W**	PHP	PM	dcl.sourceforge.net
OTRS	**WE**	Perl	PMOD	www.otrs.org
JitterBug[c]	**WE**	C	F	www.samba.org/jitterbug
WREQ[c]	**WE**	Perl	G	math.duke.edu/~yu/wreq

a. 输入类型：W=Web、E=电子邮件。
b. 后端：M=MySQL、P=PostgreSQL、O=Oracle、D=DB2、F=普通文件、G=**gdbm**。
c. 不再得到维护。

我们更喜欢 Mantis 一些。原本开发它是为了跟踪一种视频游戏软件里的缺陷（bug）。它可以在 Linux、Solaris、Windows、Mac OS 甚至 OS/2 上运行。它有轻量级、很简单、易修改和可定制的特点。它需要 PHP、MySQL 和一个 Web 服务器。但它最重要的特点是文档好！

另一种不错的系统是 OTRS（Open Ticket Request System）。OTRS 的特点在于给客户和系统管理员都提供了 Web 界面和电子邮件接口。OTRS 高度可配（例如，欢迎消息可以按队列来配置），它甚至能记录一个工单所花的时间。从 OTRS 的 Web 网站可以下载我们举例的每种 Linux 平台上的 OTRS 软件包。

RT: Request Tracker 第 3 版有丰富的功能，包括给用户和系统管理员的高度可配的 Web 界面。因为它使用了大量 Perl 模块，所以安装的时候稍微有点儿麻烦。它的命令行接口至多算是个基本型。有些站点使用打过补丁的第 1 版，既修改了 RT 更为烦人的一些缺点，又没有增加像新版本那样的复杂性。

表 30.2 给出了一些用于需求管理的商业软件选择。因为商业产品的 Web 网站大多宣传得很多了，所以没有列出诸如实现语言和后端软件这样的细节。

表 30.2 商业的故障工单系统

名 称	规 模	URL
Remedy（现在叫 BMC）	大型	www.bmc.com/remedy
ServiceDesk	大型	manageengine.adventnet.com
HEAT	中型	www.frontrange.com
Track-It!	中型	www.numarasoftware.com

有些商业产品复杂到了需要一到两个专人维护、配置和保证运行的程度（Remedy 和 ServiceDesk）。这些系统适合有很大 IT 部门的单位，但是对于人数不多、工作任务繁多的 IT 部门是一种浪费。

30.9.4 派发故障工单

在规模较大的系统管理员小组里，甚至是一个有着强大的故障工单系统的管理员组里，仍然会留下待处理的问题。让几个人分散精力，既关注当前正在从事的任务，又关注排队待处理的请求，尤其请求是通过电子邮件发送到个人邮箱的时候，这样做效率不高。我们实验了两种解决这个问题的办法。

我们的第一种尝试是让我们的系统管理员小组的成员按半天轮班处理排队的工单。在班上的人要尽可能多地回复用户发来的请求。这种方法存在问题，即不是每个人都有能力回答所有的疑问和处理好所有的问题。答案有时候不正确，因为在班上的是新人，还没有真正地熟悉客户、环境或者其支持合同的具体内容。结果，老成员还要关照一下，所以不能真的把精力集中到自己的工作上。最后，服

www.TopSage.com

务质量下降，没有取得切实的效果。

在有了这样的经验之后，我们创立了一个“分号员”的岗位，由一组高级系统管理员每月轮流担任。这个分号员负责检查故障工单系统，获得新输入的请求，再把任务分给专门的员工。如果必要，这个分号员还会联系用户获得给请求确定优先级所需要的更多信息。这个分号员使用一种自产的员工技能数据库，以此决定支持团队中的什么人有合适的技能和时间。分号员也要确保请求得到了及时解决。

30.10 灾难恢复

您的单位要依靠一个能工作的 IT 环境。您不仅要负责日常的运行，而且还必须有处理任何可预见事件的方案。为这样的大问题做好准备既影响到您总体的工作方案，也影响了您规定日常运行的方式。

在这一节里，我们会讨论各种不同类型的灾难、您需要细致恢复的数据以及灾难处理计划中的重要组成部分。

我们建议您以书面形式做一份清楚的潜在灾难分类表，您要对这些灾难提供保护。灾难不总是相同的，您需要几种不同的方案来涵盖所有的可能性。考虑下面的威胁。

- 安全泄漏：在 2000 年之前，大约 60%来自单位内部。到 2001 年，外部攻击数量激增，使得前面的百分比下降到大约 30%。到 2005 年，基于内部的攻击反弹到 47%[6]。
- 环境问题：电源的瞬时峰值和突然中止、制冷系统故障、洪水、风暴、地震、流星、恐怖分子或者外星入侵。
- 人为错误：文件及数据库的删除或破坏、配置信息丢失。
- 硬件自行崩溃：服务器停止工作、硬盘烧毁、网络故障。

问问自己，我的用户从 IT 环境中能获得什么好处？这些好处中哪些最重要？什么会让这些好处遭到风险？我们该如何保护自己不受这些威胁？获得问题的正确定义不是一个简单的任务，整个单位（而不仅是系统管理组）都要投入进来，这很关键。

30.10.1 备份和离线的信息

原因很显然，数据备份是灾难处理规划中的一个重要组成部分，我们在第 9 章更详细地介绍过备份。但是建立可靠的异地备份还不够。怎样才能很快地访问到这些备份呢？您该怎样用备件、人力以及备份数据替换已经发生故障的一台服务器呢？

我们习惯于使用网络来交流和访问文档。不过，在一起安全事件之后，这些设施可能不能用或者被攻击了。要把所有相关的联络信息和规程离线保存。要知道在哪里得到最近的转储磁带、使用什么 **restore** 命令，而不用去看**/etc/dumpdates**。

对于上述所有发生灾难的情况，您需要同时访问重要信息的在线和离线副本。如果可能，应将在线副本保存在一台独立的机器上：该机需具有相当丰富的补充工具，能提供系统管理员所需的关键环境，运行自己的名字服务器，拥有完整的本地**/etc/hosts** 文件，没有任何需要通过共享获得的文件，带有打印机等。不要使用一台老设备，那没有任何好处，灾难恢复机器应该速度快，而且应该有充足的内存和硬盘空间，这样在做恢复工作期间可以用来恢复和比较数据。这台机器需要一个完整的开发环境，从而可以对任何被攻击受损的软件打补丁和重新编译。如果这台机器还有各种硬盘接口，符合您

[6] 这个数据来自于 FBI（美国联邦调查局）和 CSI（计算机安全协会，Computer Security Institute）。

www.TopSage.com

的站点上用到的所有硬盘类型（IDE、SATA、SCSI、FC-AL 等），那么也会有帮助的。

下面是保存在备份计算机上和以打印形式保存的一份数据清单。

- 灾难处理规程大纲：需要取得联系的人、进行联系的时间、交谈的内容等。
- 服务联络电话及客户电话。
- 当地的主要联系电话：职员、警察局、消防队、公司老板、职业介绍所。
- 硬件及软件配置的数据：操作系统版本、补丁级别、分区表、PC 的硬件设定、IRQ、DMA 等[7]。
- 备份磁带及生成备份的时间安排。
- 网络地图。
- 软件的序列号、许可证数据以及口令。
- 应急盘供货商的联系信息，以备不时之需。

30.10.2 处理灾难的人员准备

应事先确定负责处理灾难事件的人员。预先建立一系列指令而且离线保存主要人员的姓名和联系电话。也许最佳人选是直接面向用户的系统管理员，而不是 IT 主管（对于这样的角色，IT 主管往往是个不好的人选）。

负责人必须是有威信和有决策权的人，他能够根据最少的信息做出艰难的决定（例如，一项切断整个部门网络的决策）。做出这样的决策，以一种明智的方式与之沟通，而且在危机中实际领导系统管理人员可能要比只有对系统和网络管理的理论知识重要得多。我们保留着一张小卡片，上面用小字体印着重要的人名和电话号码。非常方便——而且能装在钱包里。

在大多数灾难计划中非常重要，但有时又没有被讲到的前提是，必须有系统管理人员参与处理上述情况。遗憾的是，这些人员会生病、毕业、度假或者跳槽。当您需要额外的紧急帮助时最好做什么是值得考虑的（如果系统十分脆弱或者用户的水平较低，则缺乏足够多的系统管理员有时也会间接导致突发事故）。

您可能会尝试与当地咨询公司或大学建立协议，因为他们拥有可以共享的系统管理人才。当然，您必须愿意在您的这些朋友们遇到问题时也提供援助。最重要的是，不要在日常例行工作中负担太重。应雇佣足够多的系统管理员，不要期望他们一天工作 12 小时。

30.10.3 电源和 HVAC

应在使用前检测灾难恢复计划。每月或者每季度检测您的发电机和 UPS，这取决于您的管理工作情愿接受多少风险。确认您所关心的每样设备都插到了 UPS 上，UPS 电池状态良好，而且故障反应机制能起作用。把 UPS 插头从墙上的插座中拔出，即可对单个 UPS 进行检测。为了确保关键设备都有 UPS 电源保护，您可能不得不在线路中废除断路开关，并确认紧急配置与预先计划的功能相同。

UPS 也需要维护。这项工作或许是系统管理职责之外的任务，但是您要负责保证维护工作的执行。大多数电源的干扰持续时间都较短，但在事故计划中应至少设计两个小时的电池寿命，以保证在断电时间较长时，有足够的时间正常关闭计算机。有些 UPS 具有串行端口或以太网接口，利用此接口可在断电 5 分钟（该时间可配置）后关闭一些不重要的计算机。

利用电源中断的这段时间可进行任何您已计划好的 5 分钟升级过程，比如为服务器添加一块硬盘。不管怎样已经关机了，因此总会使人觉得不方便。在有些情况下，在停电期间多增加 5 分钟比用一星期时间通知一次预定的关机更容易让人接受。如果有您认为不再使用的旧机器，就关闭它们，直到有

[7] 网络探测和资产管理工具可以自动编制出许多这类数据。H-Inventory 和 LANsurveyor 是两种可以考虑的备选软件。

www.TopSage.com

人抱怨为止。可能在数周以后——或者永远不会——才会有人注意到这台“不见了的”机器。

冷却系统经常有一个报警系统，如果温度过高，这个系统可以呼叫您。适当设定“过高”保险的数值，以便在冷却系统呼您之后，系统开始烧焦之前，您有时间赶到，我们使用 76 度而不是 90 度（译者注：华氏度），因为我们的驻地在 45 分钟路程（夏季，冬季时间不固定）之外的山里。在机房内保留一个机械或电池驱动的温度计——停电也就意味着您丧失了所有那些正常情况下能够告诉您温度的电子指示器。

一家美国政府的大型实验室最近建成了一种新型机房，其中安装有运行大型科学模型的 256 节点的 Alpha 集群。所有设备都与 UPS 电源相连而且所有设施均具有最先进的设计。遗憾的是，一次小小的供电中断竟使中心陷入长达 4 个小时的瘫痪状态。为什么呢？原来是控制 HVAC（空调）的 PC 没有与 UPS 相连。这台计算机发生了故障并致使空调系统失灵。由此可见，要认真进行测试。

如果您要在一个远程地点放置设备，那么在签合同之前，要看看主机托管站点的电力设施。核实确有发电机，并做定期检测。要求在下一次检测发电机的时候到现场观摩，无论您是否能看到一次实际的检测，您都可能会获得有用的信息。

30.10.4 网络冗余性

ISP 偶尔会因为并购而被吞并。这样的合并破坏了许多公司为维护网络与 Internet 的冗余连接而认真排定的计划。合并后的 ISP 将整合属于各独立公司的网络。原来拥有独立 Internet 通路的客户，现在则可能获得运行于单一通道上的多条连接，因而又将受到可能出现的光纤挖断的影响。

我们也知道 ISP 宣传的“冗余电路”或者“备份连接”值得怀疑。凑近一些再观察，您可能会发现没错，是两条光纤，但是两条光纤都在同一管线里，或者备份连接通过一条已经饱和的 ATM 网络传输。每年复查一下您的 ISP，核实您仍然有名副其实的冗余性。

30.10.5 安全事件

系统安全的内容在第 20 章详细介绍过了。不过，这里也值得提一下，因为安全考虑影响着大部分的系统管理任务。您的站点上没有哪一项管理策略可以在不考虑安全的情况下制定。第 20 章从很大程度上集中讨论了防止安全事件发生的多种方式。但是，考虑如何从安全事件中恢复也和安全规划同等重要。

让您的 Web 网站被“劫持（hijack）（译者注：即网站被攻击后每次访问都自动跳转至另一个页面或者网站）”是一种让人格外难堪的入侵。对于在提供 Web 主机托管的公司工作的一名系统管理员来说，发生劫持攻击就是一次灾难性的事件，特别是那些处理信用卡数据的站点。电话会从客户、媒体以及由新闻获悉此事的公司副总那里蜂拥而来。那么该由谁负责接听这些电话？面对各种提问应做出怎样的回答？应由谁负责这项事务？每个人的具体分工是什么？对于一个得到高度关注的商业公司来说，事先考虑这类情况，提前准备好相应的答复，可能还会拟定具体的实施细则，这些肯定是值得的。

接受信用卡数据的站点在被劫持之后要依法处理。要确保您单位的法律部门参加到安全事件的处理策划中，并且要保证在发生危机的时候能找到相关联系人的姓名和电话号码。

当 CNN 或 Slashdot 公布您的网站陷于停顿时，会导致您的 Internet 流量激增，和高速公路上的车流慢下来观望路边发生的事故产生的效果一样，经常会达到能破坏您刚刚所做的修复的程度。如果您的网站不能处理流量增加 25%或更多的情况，则应考虑使用负载平衡软件将多余的连接传送到一台只提供一个网页“对不起，现在网络正忙，无法马上处理您提出的请求”的服务器上。

www.TopSage.com

30.10.6 世贸中心的二手故事

我们曾经听说过一些和2001年9月11日世贸大厦受攻击事件有关的故事，它们很有启发性。遗憾的是，“知道”这些故事的系统管理员不能得到其管理层的许可把它们说出去。我们没有放弃，而是在这儿写进来一些倒过两手甚至三手的故事，它们可能不是100%真实的。它们肯定不是有关灾难管理的最后故事，然而它们却很有意思。

一家ISP把它们所有的电话和网络流量都通过位于世贸群楼中一幢小楼的设备路由出去。ISP的建筑在攻击中幸免于难，在应急发电机的电力供应下，那个设备继续在运转（虽然在大楼的空调系统停止工作以后它遇到了散热问题）。遗憾的是，ISP的员工没有能到现场给发电机的油箱加油。寓意：保证至少要有几天的发电能力支撑应急状态。

另一家位于世贸双塔之一的机构一度把自己的灾难恢复设施安排在双塔的另一塔上。幸好这家公司在攻击前又重新考虑了“异地”的要求。第三家机构之所以能够恢复，是因为其所有的重要数据都在多个地点的多个设备上有备份，通过光纤通道同步镜像。不过，这家机构丧失了大量书面文档——不是所有的文档都在计算机里有。

一家位于世贸大楼上的金融公司把其异地备份存储外包了出去。不过，这家公司不知道存储设备也在世贸的另一座大楼里。知道您的异地资料在哪里保存是很重要的。

CNN的Web网站在9月11日经受的流量曾超出了CNN的负载监测器的监测能力。纽约市的电话系统也过载了，通过世贸大厦顶上的天线发送电视信号的电视台则发生了节目中断。每个人都涌向Internet，访问cnn.com。这个网站首先过载。在试图把服务器转移到各处来解决问题之后，CNN最终把它的主页的复杂性降低到就是一幅图和没有链接的文字。几个小时之后，这个网站才再次恢复，并且能够承受那么大的负载。

经历过这次大规模Web流量冲击后，CNN在主页内容的复杂性何时要降低以及谁可以做出这样的改动决定上面，改变了自己的策略。权力和责任从管理层下放给了系统管理部门。

30.11 书面政策

在本章的讨论中涉及在系统管理、计算机安全、标准化、计算机法规方面的重大问题。我们惊奇地发现，它们都提到“经签署的、书面的政策”是一个健全机构所必须具备的。

政策和规程应该写成书面形式，征得管理部门的认可，并通过律师的检查。这些准备工作最好是在用来处理棘手问题之前就已完成了。应该存在以下几种不同的政策：

- 管理性服务政策；
- 用户的权利与义务；
- 有关系统管理员（特权用户）的政策；
- guest账号政策。

复查表或处方形式的规程可用来将现有的实践方法编成法规。它们无论对新上任的系统管理员还是经验丰富的系统管理员都十分有用。标准规程具有以下几点好处：

- 总以同样的方式处理日常事务；
- 复查表形式减少了出现错误或遗漏步骤的可能性；
- 系统管理员利用处方进行系统管理的速度更快；
- 对所发生的变化自建文档；
- 书面的规程提供了衡量正确性的标准。

www.TopSage.com

现在，Linux 在企业界取代了过去的大型机承担着关键性的任务。在这类大型的 Linux 应用场合，复查表（经常又称其为“运行手册”）担当着为公共任务提供文档资料的作用。它们通常在线保存，也印刷成手册。编写并维护运行手册的往往是系统管理员，他们比使用手册的支持人员要高一个层次，但是，从长远来看，这样的组织及标准化方式会带来收益。

您可能想为之建立规程的常见任务有：

- 添加主机；
- 添加用户；
- 机器的本地化处理；
- 为新机器建立备份；
- 对新机器提供安全保障；
- 重新启动一个复杂的软件；
- 恢复不响应或不提供任何数据的网站；
- 疏通打印机的工作状态和重新启动打印机；
- 升级操作系统；
- 给软件打补丁；
- 安装软件包；
- 从网络安装软件；
- 升级关键软件（**sendmail**、**gcc**、**named** 等）；
- 备份及恢复文件；
- 备份磁带的作废设定；
- 执行紧急关机操作（所有主机，除一些重要主机之外的全部主机等）。

政策和规程间还存在一系列的问题。例如：

- 谁可以有账号？
- 当他们离开时怎么办？

这类问题的解决方案需要以书面形式记录下来，以便您可以保持政策的一致性而避免多种可能性（这就好像一个 4 岁小孩经常耍的把戏，“妈妈说不行，让我们再去问问爸爸吧！”）。通常情况下，“如果”部分为政策，而“怎么办”部分则为规程。

有些政策决定将由您所运行的软件或由外部机构（比如 ISP）的政策来决定。如果要维护用户数据的隐私权，那么需要强制执行一些政策。我们把这些政策叫做“没有商量余地的政策”。

特别地，我们认为应该在整个网点的基础上管理 Internet 地址、主机名、UID、GID 及用户名。有些网点（例如，跨国公司）的范围明显过大，不能实现上述政策，但如果您能够对其进行适当的变动，在整个网点的基础上进行管理会使事情变得简单许多。我们知道的一家公司对 35 000 用户和 100 000 台机器作为一个站点实施管理，所以一个单位变得太大，不适合做全站管理的阈值必须相当大。

影响可能要比本地系统管理小组范围更大的一些重要的政策问题有：

- 安全入侵的处理；
- 文件系统导出控制；
- 口令选择准则；
- 由于某个原因取消登录账号；
- 受版权保护的资料（例如，MP3 和 DVD）；
- 软件盗版。

在一个大型网点内的管理小组之间良好的交流渠道能够防止问题并且还有助于加深彼此的信任与合作。可将举行聚会看作是进行沟通的“交流工具”。有些系统管理小组使用类似 IRC 的 MUD 或 MOO 作为其沟通的“交流工具”。这个方法虽然废话较多，但如果使用得当会让您的机构运行得更流

www.TopSage.com

畅，尤其适用于某些人员在异地或在家的情况。聊天会被监听，所以要注意不要通过一个第三方网络发送有关网络和单位的敏感信息。

30.11.1 安全政策

用户希望在哪个方面受到保护？是数据还是硬件？还是从系统故障中迅速恢复的能力？在为网点设计安全政策时，必须对若干方面进行权衡：

- 提供服务与安全保障的关系（较多的服务＝较低的安全性）；
- 使用方便与安全保障的关系（安全性＝1/方便程度）；
- 安全成本与损失风险的关系。

RFC2196 即 *Site Security Handbook*，它是 IETF（Internet Engineering Task Force，Internet 工程任务组）下属的研究组于 1997 年编纂的一份长达 75 页的文件。它在安全性问题、用户政策及规程等方面为系统管理员提供了参考意见。它并不包括具体用于保护 Internet 站点安全的方法，但它的确能够提供一些极具价值的信息。文件的最后 15 页提供了丰富的在线及已出版的参考资料。

RFC2196 建议在政策文档中包括下面几点。

- 计算机软件和硬件的*购置原则*。由于系统管理员通常对供货商市场人员不提的硬件问题、软件缺点以及支持问题等比较了解，因此在购置过程中有系统管理员参与会有巨大成效。
- 设定有关监视用户电子邮件及按键操作的*隐私政策*，以及用于处理用户文件的政策。
- *访问政策*：谁能访问、他们能通过访问做什么、他们能安装什么样的硬件和软件等。这个文档应包括与隐私政策相同的对授权使用和连线监视的警告。
- 规定用户和系统管理员职责义务的*责任政策*。该政策应该明确说明谁可以监听网络流量，阅读用户的电子邮件，以及调查其他类似的敏感地带。它还应该概述可以进行这样调查的情形。
- 设定口令及远程访问原则的*身份验证政策*。
- *可用性政策*说明系统要何时开机，列出计划的维护时间，给出报告问题的指示，以及设定期望的响应时间。
- *维护政策*，包括外包原则以及使用第三方维护人员的规程。

RFC2196 清单中的明显疏漏是没有包括规定谁可以授权新账号和扩大特权的授权政策。最初版本的 RFC1244：*Site Security Handbook* 包含的是具体问题的清单，而不是在系统管理员看来稍微有用一点的政策类型。较新版本的 RFC 则包含了针对机器可能运行的每种服务类型的建议，并且描述了服务方面的各种问题及可能的解决方法。

ISO 117999 是一份更新的参考，在业界得到了广泛的使用。COBIT 是另一种流行的标准（这个标准的内容不止是安全）。

无论您采用的是何种政策，它们都必须具有清晰易懂的书面形式，并且经过所有用户和系统管理员签署。落实起来必须保持一视同仁，甚至当用户是为计算服务而付费的客户也不例外。落实政策不能一视同仁将会削弱它们在法律及理解方面的有效性。

30.11.2 用户政策协定

在科罗拉多大学的计算机科学系，用户政策是以一个初始 shell 的形式呈现给用户的，这个 shell 显示出政策，而且要求用户在他们能得到一个真正的 shell 并使用其账号之前同意并“签署”该政策。这一方案不但节省了时间，而且还可以避免产生分歧，但在实施该方案之前，应先征求您的律师的意见。

下面是用户政策协定应该明确规定的一些问题：

www.TopSage.com

- 与朋友及亲戚共享账号（我们建议：绝不做）；
- 对本地 passwd 文件运行口令解密程序；
- 对其他站点的 passwd 文件运行口令解密程序；
- 破坏服务；
- 侵入其他账号；
- 误用与伪造电子邮件；
- 电子邮件和电子介质的一般性使用；
- 查看其他用户的文件（如果可读、可写或被邀请分别怎么办？）；
- 从网上下载软件（从不？总下？如果用户检查无误分别怎么办？）；
- 强制在 Windows 主机上使用防火墙和防病毒软件；
- 使用系统资源（打印机、磁盘空间、调制解调器、CPU）；
- 复制具有版权的资料（软件、音乐、电影等）；
- 允许其他人复制有版权的资料；
- 将资源用于私人或者商业活动；
- 从事非法行为（欺骗、诽谤等）；
- 参与在某些地方违法而在其他地方不违法的活动（如色情文学、政治活动）。

但是我们应该认识到，没有哪个政策协议能防止得了有人破坏协议。您可以把它当作一种武器，打击犯罪、删除其账号、解雇或驱逐违法者，或者在您的环境里任何合适的措施。

我们在这里提供了墨尔本大学计算机科学系学生在使用学校提供的计算机前必须签署的协议，作为一个简短政策协议的例子。

我，签名人，特此声明将严格遵守以下的各项规定。

- 本人把本系的计算机及网络设备只用于与计算机科学专业课程学习直接相关的目的。
- 本人尊重本系的计算机账号只能供接受人本人使用的规定。因此，我不会授权或帮助他人使用本人的账号和文件，也不会将本人的登录口令告诉给其他任何人。
- 当没有得到正确和清楚的授权时，本人决不访问或试图访问任何计算机、计算机账号、网络或文件。上述访问违反了州和联邦法律，并且与本校校规相抵触。如果本人发现上述情况，将立即通知计算机科学系办公室。
- 本人了解我可以在文件系统中访问的某些软件及数据是受版权和其他法律以及许可证和其他协约保护的。因此，本人不会违反这些规定。
- 本人决不利用本校的设施获取、制作、使用或传播未经授权的软件副本。
- 本人将严格遵守关于本校授权使用软件的机密（包括其中使用的方法和概念），同时本人也决不将自己许可范围内的技术泄漏给他人，以避免给学校造成不必要的损失。
- 本人以人格尊严保证诚实文明地使用本系计算机及网络设备。本人进一步保证避免任何与使用本系计算及网络设施有关的活动给本系或本校的声誉带来任何影响。

本人将严格遵守墨尔本大学校规 8.1.R7（参见学生守则）中有关本校计算机及网络设备使用的条款。

如果有任何违反上述规定的行为，本人甘愿接受处罚。包括作业和课程作不及格处理、暂停或取消对本校计算设施的访问、停学或开除本校学籍、罚款和/或根据 1988 年刑法（计算机部分）给予相应的处罚[8]。

这里应注意诚实、人格尊严以及不给本校声誉造成损失等说法。此类模糊的要求将为事后处理提供了一定的调整空间，并且有助于弥补政策中的不足之处。尽管它们真正的法律效力可能被忽略了，

[8] 要记住，虽然在美国已经通过了类似的计算机和软件的相关法律，但这里却是一项澳大利亚的法律。

www.TopSage.com

TopSage.com

但在您的政策协议中保护这样的要求仍不失为较好的表达方式。

30.11.3 系统管理员政策协定

用于系统管理员（及其他有特殊情况的人）的政策文档中必须设立使用 root 特权以及尊重用户隐私权的指导原则。IT 经理必须确保系统管理人员理解并且维护这样一份文档的含义。同时，对于用户抱怨尚未看到被退回的邮件的内容它就已损坏，很难做出适当的回复（但是保留一份邮件消息头的副本往往足以说清问题的特征并解决它）。

如果您的网点使用了诸如 **sudo** 这样的工具来进行 root 访问，系统管理员必须利用不易破解的口令，而且不与任何人共享登录账号。可以考虑定期对系统管理员的口令运行 John the Ripper 程序。必要的一点是，系统管理员不执行 **studo sh** 或者在 **sudo** 其最喜欢的文本编辑器之后不转入 shell，这些都是装样子的 **sudo** 用法，会使 **sudo** 的日志记录功能生效。日志不是用来暗中监视系统管理员的，它是在出错的时候用来再现连续发生的事件的。参见 3.4.2 节了解有关 **sudo** 的详细内容，参见 20.10.3 节了解有关 John the Ripper 的更多资料。

对于某些系统管理员来说，想要炫耀超级用户权限的急迫心理超过了常识。可以礼貌地建议他们改行。

在一些网点，拥有 root 的口令是一种身份的象征，这可能比经理盥洗室的钥匙更有价值。通常，拥有这些口令的人是不需要该口令或不应该有该口令的工程师。我们知道有一个网点可以给所有的工程师都提供 root 的口令，但又规定所有拥有口令的人员必须配备传呼机以便及时为他人提供帮助。结果想要口令的人数大减。

另一项我们成功使用的技术是把 root 的口令封装在信封中并将其放置在系统管理员知道的一个地点。系统管理员一般用 **sudo** 做他们的工作，但他们出于某种原因确实需要 root 口令时，他们可以开启这个信封。随后，root 口令就被改变并放回一个新的信封。这个规程相当简便，但因为要鼓励使用 **sudo**，所以仍会让人不快。重要的是，在推新口令的时候，那些宕机或者离线的机器的老口令要保留一段时间。

如果您的员工知道 root 口令，那么一旦该员工离职，那么就必须修改这个口令。在您的站点上可能有多个管理员口令：计算机的、打印机的、网络硬件的、PROM 或者 BIOS 的、数据库的等。如果有员工离职，那么把必须要改的（以及如何去做）都写下来。

30.12 法律问题（美国）

美国联邦政府和几个州都有涉及计算机犯罪的法律。在联邦法律（美国）中，有两项是在 20 世纪 90 年代初期通过的，而在最近又通过了两项：

- 联邦通信隐私权法（Federal Communications Privacy Act）；
- 计算机欺诈和滥用法（Computer Fraud and Abuse Act）；
- 禁止电子盗窃法（No Electronic Theft Act）；
- 数字千年版权法（Digital Millennium Copyrigh Act）。

法律上的几项重大问题有：系统管理员、网络操作员以及 Web 主机托管站点负有的责任，电子商务的强加密，P2P 网络及其对娱乐业的威胁，版权问题，隐私问题。本节的这些专题对上述问题以及系统管理有关的其他各类法律问题给予了说明。

30.12.1 加密

在进行电子商务活动和电子信息交流时需要加密，这是很显而易见的事情。不过，加密与某些国

家的法律相抵触。执法机关不想让公民能保存他们（即警察）无法解密的数据。

在美国，有关加密的法律也在不断进行修改。过去出口任何形式的强加密技术都是犯法的。软件公司必须同时制作加密软件的两个版本：一个在国内市场销售，而另一个经过削减的版本则用于出口。这项政策的副作用之一就是许多与加密有关的软件项目不得不以其他国家为基地。这项政策明显的荒诞之处（其他国家在更早的时间就已拥有加密技术）以及电子商务的要求最终迫使政府不得不改变姿态。

尽管出口限制尚未彻底消除，但 2004 年引入的修正案让美国的情况要比以前好得多。

IETF 已经在端到端安全通信领域的协议层上制定了标准——IPsec 标准——而供应厂商也随即着手发售带有该协议的系统。身份验证部分一般是绑定的，但加密部分经常要单独安装。这种结构为不能使用加密技术的国家保留了灵活性。

在一次最近（2006 年）的美国法庭判决中，软件代码被认为就像言论自由一样，被判受到美国宪法第一修正案的保护。而在 1990 年，Dan Bernstein 当时还是 U.C. Berkeley 的一名研究生，他开发了一种新的加密算法。不过，美国外交部不允许他在公开场合讨论这个算法，因为这个算法被归为“军火”一类的技术。EFF（电子先锋基金会，Electronic Frontier Foundation）追查这个案子最终获胜。只用了 16 年时间！

30.12.2 版权

音乐和电影工业界已经有点儿惊恐万状地发现家用计算机能够播放 CD 音乐以及 DVD 电影。这对于它们来说既是一种机会也是一种威胁。最终的结果则取决于它们二者对此的反应采取的是积极还是消极的方式，可惜他们似乎选择了消极反对的路子。

DVD 格式采用 CSS 技术（即 Content Scrambling System，内容加扰系统），通过加密密钥对碟片内容进行编码。上述方法主要是为了限制 DVD 只能用得到许可和批准的播放机播放。消费类的 DVD 播放机包含适当的解码密钥，这类似于大多数 DVD 计算机驱动器随带的软件播放器。

一名挪威的学生采用逆向工程的方法回溯了 CSS 加密过程，并在 Web 上张贴出一个叫做 DeCSS 的程序。该程序并没有绕过 DVD 加密方案，它只是使用一个合法的来自于 Windows 播放器的解码密钥来对 DVD 的数据流解码，并将结果保存在磁盘上，所以他能在 Linux 上播放 DVD，而不必使用 Windows。

美国电影联合会（Motion Picture Association of American，MPAA）和 DVD 拷贝控制联合会（DVD Copy Control Association）都对采用 DeCSS 软件的大量“发布人”提起了诉讼，有 Web 网站链接到一份 DeCSS 拷贝的每个人都被认为是“发布人”。诉讼中指控被告不但盗窃了版权，而且涉嫌传播商业机密并“规避拷贝保护”[9]。根据美国 1998 年颁布的数字千年版权法（Digital Millennium Copy Act，DMCA），后者的行为被重新定为一种违反版权的形式。MPAA 打赢了这场官司，DeCSS 被认定违反了 DMCA。目前（2006 年），在美国没有能合法使用的开放源代码 DVD 播放器。

在最近的两个案子里，一些公司已经尝试使用 DMCA 法中有关版权和反逆向工程方面的内容来遏制其竞争对手。Sears 公司禁止将第三方的车库门遥控器用于操作 Sears 品牌的车库门，而 Lexmark（利盟）试图用法律禁止其他公司重新填充利盟的空激光打印机墨盒，并以比新墨盒更低的价格销售。在这个两个案子里，EFF 都以弱胜强。

另一件 DMCA 的案子是 MGM 诉 Grokster 和 Streamcast，这个案子打到了美国最高法院。以前的裁决（例如，20 世纪 70 年代的索尼 Betamax 录像带一案的先例）都认定，销售能够破坏别人有版权

[9] DMCA 禁止出于保护数字内容安全而采取的“规避”手段，即使规避的理由是在传统版权法所说的“合理使用”免责范围内也不行。不用说，这个变化象征着大大剥夺了合理使用免责范围。

www.TopSage.com

的东西的技术不违法，您必须用该技术实施破坏才算违法。在 Grokster 的案子里，Streamcast 网络公司依靠宣传其软件能够复制有版权的音乐和电影来赚钱。这一引诱别人破坏版权的行为被判违反了 DMCA，法庭 9 比 0 支持 MGM，认为 Grokster 有罪。

P2P 网络（如 BitTorrent）可能不会受到这个“引诱别人违法”理由的指控，因为许多实现文件共享的应用软件都不包括有版权的资料。不过，细节仍然不明朗。

SCO 已经彻底调查过 Linux 提供商，掌握其是否在自己的发行版本中使用了一点点有版权的 UNIX 代码。他们起诉过成千上万的个人，还起诉过 IBM。ODSL（Open Source Development Labs，开放源代码开发实验室）设立了一个 1000 万美元的法律基金来保护 Linux 用户。其中一些支持 OSDL 的公司包括思科、戴尔、爱立信、富士通、日立、IBM、英特尔、诺基亚、Red Hat、Sun 和东芝公司。最近（2006 年 7 月）的法庭判决支持 IBM 的辩护，认为 SCO 公司超过 250 项的指控大多数太普通了，应该去掉。判决的措辞是不鼓励起诉，所以说可能 SCO 的官司碰到了障碍，愚蠢的行为将会停止了。

30.12.3 隐私

保护隐私一直是十分困难的事情，而 Internet 的蓬勃兴起使这个问题变得比以往更危险了。以最近发生在密执安大学（University of Michigan）的事件为例，该校医保系统内存放的病人医疗记录被意外地公布在 Internet 上。在一个学生注意到这个疏漏之前的数月里，这些数据可以随意获得。充满信用卡号的数据库也被攻破了。几乎每天都会收到冒充是从您的银行发来的电子邮件，说您的账号出了问题，要求您核对账号数据。仔细看一下电子邮件的文本，而不是让它按照 Web 网页的样子呈现，就会显示出数据会被送给东欧或者亚洲的黑客，而不是您的银行。

一、DoubleClick

另一个有关隐私的大丑闻是一起故意的事件，它涉及了 DoubleClick（一家提供多种网页广告栏的广告代理商）。DoubleClick 多年来始终声称其系统中的用户永远不会被跟踪或被识别出来。但最近 DoubleClick 购买了一家从事数据挖掘的公司，而且开始从每个访问包含 DoubleClick 广告的网页的用户那里收集数据。随后激起的众怒致使 DoubleClick 公司不得不取消该项计划，并且雇佣了两名资深律师专门以“隐私”的立场来寻求一种能让 DoubleClick 合法地了解哪些用户易受其广告影响的途径。

二、索尼的 rootkit

索尼在数十种音乐 CD 产品上加入了一个叫做 XCP（代表 eXtended Copy Protection，延伸拷贝保护）的程序。这个软件把它自己安装在用户的计算机上，并且把自己的文件隐藏起来让用户不会注意到，然后把用户在听的音乐报告给索尼公司。因为 XCP 是一种 rootkit 攻击的形式，所以安装的文件会把计算机暴露给黑客，参考 20.9.10 节。

Mark Russinovich 从 amazon.com 购买了带有 DRM（Digital Rights Management，数字权利管理）的索尼音乐 CD，把它安装到自己的 Windows 系统中。随后，他发现了索尼的 rootkit，并且分析了它的影响和范围。

“索尼公司在我的系统上安装的软件使用了通常只有恶意软件为隐藏自己才会采用的技术，而且这个软件写得还很差，没有办法卸载。更糟的是，大多数用户通过 rootkit 扫描工具偶然发现被隐藏的文件后，如果他们尝试用直接的方法删除这些隐藏文件，那么就会导致计算机出问题。

虽然我理解传媒界有权使用拷贝保护机制来防止违法拷贝，但是我并不认为我们找到了在合理使用和拷贝保护之间的正确平衡点。索尼把 DRM 做得过头了，这就是一个明证。”

存在 rootkit，而且不能卸载的事实在 EULA 许可证中并没有提及，用户必须签署 EULA 才能在他们的计算机上安装 CD[10]。在英国，这种行为明白无误地违反了他们的计算机滥用法案。在加利福尼亚，

[10] EULA 代表软件终端用户许可证协议（End-User License Agreement）。

www.TopSage.com

根据加州商业和保护法这种行为也是不合法的，要处以每台被感染的计算机 1000 美元的罚款。EFF 已经开始联名起诉索尼。

系统管理员为什么应该留心这个？如果您的桌面 PC 安装了索尼的 rootkit，您的整个站点也就有可能被入侵。

三、调用记录和 Web 日志

美国最大的三家电话公司遭到指控，内容是它们把自己 2 亿个用户的电话和 Internet 记录提供给国家安全局（NSA，National Security Agency）。另一家美国的大型电话公司 Qwest 因为隐私问题，拒绝同 NSA 合作。当 NSA 强烈暗示 Qwest 可能再也拿不到保密的政府合同的时候，Qwest 也没有退让。Qwest 好样的，EFF 正在起诉其他三家电话公司。

历史已表明电话记录可能会被滥用。2002 年，一台哥伦比亚贩毒公司的计算机被执法机关抓获。在对这台计算机上硬盘里的数据进行分析时，暴露出在 1994 年，电话交换机的电话记录就已经用来判断贩毒公司的职员与执法机关有没有“关系”了。几名雇员因为“太过密切”而被杀害。流量分析是一种功能强大的工具，它正在被加以利用。

大型搜索引擎和门户网站（例如 Yahoo!）中有几家过去给美国政府提供过用户搜索 Web 情况的数据。Google 是唯一一个不向政府要求屈服的公司。

美国移动通信网络也引发了一些隐私方面的问题。当您把手机放进口袋，带着它在国内到处跑的时候，这部电话就会自动在本地接入，让电话能够接到您这里。遗憾的是，政府不需要一个正当的或者甚至可能的理由就能跟踪您的手机的位置。

其背后的法律问题就是所谓的“电话记录仪（pen-register）”数据的定义。在电话的环境下，这个数据是何人何时呼叫何人的信息，和实际的通话内容没有关系。在 1984 年，电子通信隐私法案（Electronic Communications Privacy Act）把电话记录数据和内容（两者之前都得到保护）做了区分，声称电话公司根据一种“容易获得”的法庭庭谕透露电话记录数据也是合法的。在 2001 年，爱国者法案（Patriot Act）对“电话号码”的定义做了扩展，包括进了软件程序和 Internet 相关的信息，如 IP 地址和 URL。

因此，您所浏览的 URL，以及您所执行的 Web 搜索现在都被归为“电话记录”数据。它们明确规定不作为隐私信息予以保护。在系统管理员必须交出电子邮件或者用户文件之前，要获得法庭批准令，但是大多数日志文件的内容只要求一份法庭庭谕。

“电话记录”数据的另一个特性是，即使由执法警官非法获得，也能用于审判程序。可怕的时代。

30.12.4 EULA

EULA 是 End User License Agreements（软件终端用户许可证协议）的缩写，当您安装已经购买的一个软件时，就会要求您接受该协议。大多数人都不会仔细阅读它，或者如果他们确实读了这些协议，他们也得点 Yes（同意），因为他们已经为软件付了钱，打开了 CD 包装，而且开始了安装过程——把软件还回去退钱已经太晚了。有些 EULA 就是彻头彻尾的无耻，下面是一些可能要求您同意的条款：

- 允许厂商监视或者审计您的计算机（Red Hat！）；
- 不会批评产品或者厂商（微软）；
- 不对产品进行对比评测，或者不交流评测结果（McAfee）；
- 放弃要求公司实施所声明的保护隐私政策的权力（McAfee）；
- 不能定制或者修理您的计算机；
- 允许自动软件升级；
- 不对该产品做逆向工程；
- 不从计算机上删除该产品；

www.TopSage.com

- 不和竞争者的产品一起使用该产品。

还有太多太多，我们可以肯定还少了许多……

30.12.5 政策落实

某人X可能涉嫌某项非法活动Y，对此的怀疑可利用日志文件予以澄清，但此类文件在法庭上不能作为证据。因此保护自己最好的方法是保存书面的政策。日志文件有时会带有时间戳，这非常有用，但除非您的计算机运行NTP（Network Time Protocol，网络时间协议）保证其时钟与实际时间同步，否则该时间戳也不能用作证据。

在起诉某人的误用行为时需要用到安全政策。它应该包括诸如以下形式的内容：未经授权使用本校的计算机系统不仅违反了校规而且违反了州与联邦法律。未经授权的使用行为属犯罪行为，有可能受到刑事和民事处罚，我们将依法提起诉讼。

我们建议您把一条警告内容放入**/etc/motd**（每日消息文件）中，用来告诉用户您的监视政策。我们的警告内容为：

```
Your keyboard input may be monitored in the event of a real or perceived security incident.
```

您可能想要通过在给用户的启动文件中包含通知的做法，来确保用户能至少看到一次。如果您要求使用**ssh**登录进系统（而且您也应该这么做），那么要配置**sshd.conf**一定显示**motd**文件。

确保规定用户使用了账号就表示同意您的书面政策。指明用户从哪里可以得到更多的政策文档副本，并将一些重要文档张贴在相应的公告栏上。其中还要包括对不服从者的特定处罚（删除账号等）。

30.12.6 控制=义务

ISP一般都有一份由其上游ISP指定的AUP（appropriate use policy，合理使用政策），而且也要求其下游的客户履行这个AUP。这种“自上而下”的责任制把对用户行为所负的责任分摊到用户自己身上，而不是该ISP或者该ISP的上级ISP。这些政策一直用于控制垃圾邮件（即不请自来的商业电子邮件），以及当客户在其账号里保存非法或者有版权的资料时保护ISP不受惩罚。检查您所在地区的法律，各地情况各有不同。

假如从您的网点用电子邮件发送或者向Web张贴了下流内容。如果您就是CompuServe（现在是AOL的一部分），那么这就成了一个问题。“Cubby诉CompuServe”一案就是由张贴攻击性内容引起的。法官判定CompuServe无罪，但是认为攻击性内容所在的新闻组的主持人失职。您试图控制的信息越多，那么所承担的责任就越大。

从法律的角度看，您对用户使用Internet行为监视得越多，您对他们的行为或张贴内容所负有的责任就越大。如果您知道了某些违法或应受指控的行为，就有法律责任调查它并报告您的发现。

由于这个原因，一些地方对日志数据、日志文件保存的时间长度，以及历史的日志文件在备份磁带中保留的数量都进行了限制。某些软件包（如Squid的Web高速缓存）通过在其中包括各级能帮助系统管理员调试问题而又不破坏用户隐私的日志来实现这一政策。

系统管理员应该熟悉相关企业或者大学的所有政策，而且应该确保这些政策都得到了遵守。从实施的和法律的角度来看，不正确地和不一致地实施政策比没有实施政策更糟糕。

30.12.7 软件许可证

有些网站购买了一个软件包的K个副本，而其日常工作需要使用N个副本，这里K<<N。处于这

www.TopSage.com

种情况对公司是极为不利的，可能公司蒙受的损失要远远多于购买 N 减 K 个软件许可证的投入。另外还有一些网点收到某种昂贵软件包的一个演示版本并在随后进行了破解（重置计算机上的数据、查找许可证的密码等），使其在演示期限后仍然能够继续使用。作为一名系统管理员，您该如何处理上述违反许可协议的请求和在没有许可证的计算机上复制软件的情况呢？当您发现自己所负责的计算机正在运行盗版软件时应该采取何种措施？对于从未付费的共享软件又该如何？

这是一个很艰难的话题。把没有许可证的软件删除或者为其购买许可证的提议通常不会得到公司管理层的支持。往往是系统管理员签字同意在某个时间后删除演示版软件，但管理者却决定不删除该软件。

即使对现有工作十分满意，从业者的个人修养和职业道德仍将是统一的。幸运的是，今天的就业市场中对高素质系统管理员的需求极大，找工作并不需要花费很多的时间。我们知道有这样的情况，系统管理员的直接上司不仅不处理当前的状况，同时还要求系统管理员不要落井下石。随后系统管理员给老板写了一份备忘录，请求纠正当前的情况，并记录有许可证的软件数量以及投入使用的软件数量。管理员引用了许可证协议的一些条款，并抄送给公司总裁及高层管理人员。一种情况是规程生效并解雇了系统管理员的上司。另一种情况是在更高管理层拒绝正确行事的情况下，系统管理员辞职。在这种情况下，无论您怎么做，都要书面记录下来。要求获得一份书面的答复，或者如果你们都是通过谈话，那么写一份简短的备忘录，记录下您对指示的理解，然后把它发给负责人。

30.12.8 遵守法规

2002 年，在安然（Enron）和世通（MCI）这样的公司里突然爆发的著名财务造假丑闻促成通过了一项称为“萨班斯-奥克斯利法案（Sarbanes-Oxley Act，俗称 SOX）”的美国法律。SOX 法案规定了在美国国内公开贸易的公司的审计制度，要求企业管理层的个人要为其财务数据的正确性负责。

既然计算机负责跟踪大多数公司的记录，那么 SOX 法案就会给系统管理带来影响。SOX 颁布的一些规定很简单（电子邮件日志、审计记录、内部控制等），但是有些却困难了一点儿。例如，SOX 要求您对一个数据存储库（例如，数据库）的正常活动做基准判断，创建工具检测异常的行为或者访问。您还必须做好一切活动的文档，证实您正常使用着您所实现的控制措施。法规给生产效率和系统管理的热情造成了很大的影响。

另一方面，如果您销售存储设备或者记账服务，那么萨班斯-奥克斯利法案就是一个重大利好。对于大公司来说，遵守法规的平均成本达到每年四百万美元，而且这是个不断产生的费用。因为经理人直接对违规负责，可以被送进监狱，所以他们有时候会以遵守 SOX 法案的名义，要求购买新奇的系统管理方案。

SOX 也影响到了私人和非盈利性的公司。对于打印和电子的文档来说，有书面的政策协议规定数据保存多长时间，何时进行定期清理就变得非常重要了。

有一位系统管理员在一家国际会计公司的指导下，对大型计算机公司做了一些渗透测试。他的实验证明，几乎任何人都能在不被发觉的情况下，以 root 身份进入一家公司的 Oracle 财务数据库，干预公司的账目——这正是遵守 SOX 法案所要保证决不可能发生的事情。这家会计公司要这位系统管理员修改客户的财务数据，以此证实他的观点。但是他拒绝了，可这家公司还坚持要求，于是在经历了一番思想斗争之后，他决定把这家会计公司的事儿报告给客户的企业道德办公室。最后这家会计公司被解除了合同，而那位系统管理员也失去了自己的工作。

现如今，IT 审计和管理是个大问题。对是否遵守法律进行确定、衡量和认证的规范和类似标准产生了大量的缩写词：SOX、ITIL、COBIT 和 ISO 17799 只是举得出来的几个。遗憾的是，这种字母汤在系统管理员的嘴里却留下了不好的味道，而且目前也缺乏能实现最近法律所要求的全部控制措施的软件。

www.TopSage.com

30.13 软件专利

最开始，专利局规定不能获得数学定理的专利。之后定理变成了一个算法，它仍然不能取得专利。于是算法用硬件实现，这就肯定能获得专利了。

固件可能可以获得专利。软件仍然不能获得专利。但是对专利可以上诉，下级法院之一喜欢软件专利。专利局开始违背初衷批准专利，有些情况下甚至要为 10～15 年前的应用软件批准专利。

遗憾的是，美国专利局以前对于软件缺乏全面及时的了解，批准了许多不恰当（有人会说是愚蠢）的专利。Lempel-Ziv 数据压缩算法有 5 个不同的专利。这个算法发表在一份数学期刊上，由伯克利 UNIX 实现和发布。include 文件的概念有专利。光标的概念有专利。减法作为一种对软件 Y2K 兼容问题的修正也取得了专利。把一幅图像从内存复制到屏幕窗口上的过程，和使用 XOR 操作处理重叠窗口一样，获得了专利。几种数据加密标准取得了专利。在用户界面上嵌入广告材料的概念也有专利。

根据美国的专利法规定，如果某种发明可以申请专利，那么对这种发明的每一项微小的改进也能取得专利。在软件的问题上，一切都可以看作是在专利意义上的前进一步。美国专利和商标局在软件上的确小儿科，他们始终不能判定什么是在先技术（prior art）或者非创造性（obviousness），造成他们每年批准成千上万件不严格的软件专利。

在美国甚至出现了更大的威胁：商业方法软件专利。公司游说专利局，让实际上所有可以计算机化的标准商业流程成为可以受专利保护的发明。而且诸如在一个客户找咨询台时把其账号从计算机数据库里取出这样平淡无奇的活动，都已经取得了专利。Amazon.com 在“一次点击技术”上取得了商业实践专利；Amazon.com 获得了法院的禁令，要求 Barnes 和 Noble 让他们的客户在买书的时候至少要点两次鼠标[11]。

与此相反，欧盟在 2005 年 7 月规定，任何由“用于执行思维、玩游戏或者做生意的方案、规则和方法以及用于计算机的程序”所组成的东西都不是发明，因此不能取得专利。这项表决以 648 票对 14 票取得大胜。

美国专利局正在尝试修改法律，但是危害似乎已经在许多案例上造成了。一个标志性事件就是我们记忆中 1994 年的一项属于 Compton's New Media 的专利，该专利涉及检索保存在 CD-ROM 上的数据的系统。有些分析师认为它足以覆盖现在所有 CD-ROM 产品的 80%，虽然这可能有些夸大。最终，通过软件厂商方为展示现有的在先技术而在官司上投入大量的时间和金钱，Compton's New Media 的 41 项权利要求被逐条驳回。

在先技术的发现的确是专利局处理过程中的薄弱环节。专利申请要保密，而专利局的软件专家又少得可怜，他们很难知道哪些软件真正代表新技术。诉讼最终由每件案子都最后获胜的律师决定。

在美国的法律体系中，即使专利局不给一项发明申请批准专利，也制止不了和专利有关的狡诈行为闹到法庭。NTP, Inc.在范围很广的无线电子邮件领域内申请 5 项专利。专利局全都拒绝了。5 项中有两项为终审，不可能再提出诉求。然而根据这 5 项名不副实和被拒绝的专利，NTP 居然起诉加拿大公司 RIM（Research in Motion），后者是能收电子邮件的手机 BlackBerry 的制造商。

RIM 显然处于强势位置。他们的工作完全靠自己的研发，NTP 的专利申请被专利局给驳回了。然而，听取案情的法官似乎倾向于为 NTP 发出一项禁令，认定 RIM 在美国销售新产品 BlackBerry 为非法。更重要的是，禁令让现有的客户不能继续在他们的 BlackBerry 手机上使用电子邮件！

RIM 判定这些禁令的风险太大了，他们采用庭外和解的方式，用一次性支付 612 500 000 美元的方式，换回了一份永久使用 NTP 那 5 份被拒绝的专利申请所涵盖的技术的许可证。

[11] 或许 DoubleClick.net 可以用申请“两次点击”的专利给他们决定性的一击……

www.TopSage.com

30.14 标准

标准化过程在有些情况下能帮助我们（例如来自不同制造商的 Wi-Fi 接口能够相互通信），而在有些情况下却害了我们（比如 OSI 网络协议，使数百万美元的软件工程费付之东流）。标准委员会应该将已有的事实编纂形成标准，而不是发明标准。

标准的目的是消除不平等，让消费者有可能购买来自竞争厂商的兼容产品。卷入标准化进程的一些参与方确实是想制订现有事实的标准。而另一些参与方则抱有更多的政治目的：拖延竞争对手的时间或者减少让自己的产品符合规范所需的工作量。

政府机构经常是基于标准的系统和应用的最大采购方。标准的使用允许他们能够有选择地进行购买而不必局限于特殊品牌。不过，也有些批评者把标准称为非金融的贸易壁垒——各公司拖延标准化的进程直至其产品能够跟得上标准。

标准化组织有好几家，其中既有正式的，也有非正式的。每一组织在成员资格、选举和权利方面都有不同的规则。从系统管理员或者网络管理员的角度来看，最重要的标准化组织就是自由标准组织（Free Standards Group）的 LSB（Linux Standard Base，Linux 标准库）、POSIX（发音为 PAHZ-icks，Portable Operating System Interface，可移植操作系统接口）以及 IETF（Internet 工程任务组，在 12.1.2 节介绍过）。

30.14.1 LSB：Linux 标准库

自由标准组织（Free Standards Group）是一个促进开放源代码标准的非盈利性机构。它尤其注重 Linux 的标准化工作。目前有大约 20 家企业参与这项工作，其中包括像 HP、IBM 和 SGI 这样的大型制造厂商。像 Red Hat、Novell（SUSE）、Debian、Mandriva 和 Turbolinux 这样的 Linux 主要发行商也参与其中。该组织的目标就是要让第三方的软件开发人员更容易为 Linux 编写软件产品，而且要让它们成为一种产品，而不是同样产品的 20 或者 50 个稍有不同的变体。

标准化工作所细分出的种类包括：

- 库接口，包括常用的共享库；
- 配置文件；
- 系统命令；
- 常用软件包的格式和安装指南；
- 系统接口的 API；
- 文件系统层次结构标准。

在撰写本书（英文版）的时候，LSB 3.1 在 2006 年中发表了，它融入了 ISO 的标准 LSB 核心（ISO/IEC 23360），并且首次包括了桌面功能的说明，为 Gtk 和 Qt GUI 工具包制订了标准。

至今还没有厂家发布一款符合 LSB 的应用软件——一个都没有。

30.14.2 POSIX

POSIX 是 IEEE 的一个分支，获得其标准拷贝要收取费用。POSIX 在过去几年里一直在从事为 UNIX 制定一套通用标准的工作。他们在过去的主要努力着重放在了命令和系统调用（或者说库接口）上。Linux 服从 POSIX，并且遵守 POSIX 标准。

www.TopSage.com

30.14.3 ITIL：信息技术接口库

对于Linux来说，ITIL（Information Technology Interface Library）是一个过程的标准而不是一项技术标准，但是它是一个适用于较大IT单位的标准。

上溯到20世纪80年代，ITIL系统在英格兰出现，帮助在大型站点管理大型计算机和软件开发。其重点在过程而不在人，所以单位内的工作任务可以持续执行，而不论是否有人辞职或者休假。ITIL系统确定了12个功能领域，能够很好地同大多数欧洲公司对IT的高级需求对应起来（灾难恢复、修改控制、容量规划等）。

在Internet让IT成为大多数单位里都必须要有的一个部门之后，ITIL就盛行起来。大单位需要让其IT过程有一定的结构，而ITIL不会比其他组织方式更差。每个ITIL领域都规定了最佳的业界实施方法。

为了方便工作，ITIL把每个IT运行事务或者活动称为一次“改变”、一次“事件”（某种意料之外的事务）或者一个“问题”（要长期解决的东西）。

现在有一个类似于ITIL的ISO标准（ISO 20000），所以您偶尔会看到广告说自己达到了ITIL或者ISO标准。一般而言，ITIL/ISO模型不一定适合于您的单位，您可以请一家咨询公司帮助您定制一下。30.9.4节提到的几种商业的故障工单系统声称部分遵循ITIL。

30.14.4 COBIT：信息及相关技术控制目标

在美国运作的许多大企业都寻求ITIL来帮助它们解决其自身数据库和系统管理与“萨班斯-奥克斯利法案”有关的问题。参考30.12.8节了解有关“萨班斯-奥克斯利法案（Sarbanes-Oxley Act，SOX）”的更多资料。COBIT（Control Objectives for Information and related Technology，信息及相关技术控制目标）和ITIL一样，是一种信息管理的框架，并且以业界的最佳实践为基础。COBIT的宗旨是“为商业管理人和审计人日常使用，研究、开发、宣传和推动一组权威、最新且国际化的得到广泛接受的信息技术控制目标”。

COBIT是ISACA（Information Systems Audit and Control Association，信息系统审计和控制联合会）和ITGI（IT Governance Institute，IT管理协会）的一个成员。

这个框架的第一版于1996年公布，我们现在看到的是2005年公布的第4版。最新的版本受到SOX要求的很大影响。它包括34个高级目标，这些目标覆盖了分为4个领域的215个“控制目标”：规划和组织、获取与实现、提供与支持以及监视和评估（这不是八个领域吗？）。

30.15 Linux文化

在20世纪的80年代和90年代里，很难区分出Mac用户和PC用户谁更执着。PC用户认为他们的计算机是一种有用的工具，而Mac用户则对它们的机器充满了深情。Mac用户的计算机就是家庭的一员，就像家里最喜欢的宠物一样。

曾经迷漫Mac世界的那份执着现在在Linux界里变得更加强烈了。Linux的用户不只是喜欢他们的系统——他们乐意为了保护它们、修正它们以及让它们变得比Windows机器曾经梦寐以求的理想还要更好、更快和更安全而奋斗。人们对Linux注入的热情以惊人的速度在增加，Linux文化已经有了它自己的伦理、神话和英雄。

IETF的愚人节RFC之一——RFC1149，*A Standard for the Transmission of IP Datagrams on Avian Carriers*（通过鸟类的承载传输IP数据报的标准）——引起了来自挪威卑尔根的卑尔根Linux用户组

www.TopSage.com

（Bergen Linux User Group）的一群 Linux 爱好者的兴趣。这个 RFC 定义了信鸽承载 IP 协议（Carrier Pigeon Internet Protocol，CPIP），而这群 Linux 爱好者用邻居的一群鸽子实现了这个协议。下面是他们对自己的协议实现所做的一次测试（节选自 blug.linux.no/rfc1149 项目站点）：

```
Script started on Sat Apr 28 11:24:09 2001
vegard@gyversalen:~$ /sbin/ifconfig tun0
tun0    Link encap:Point-to-Point Protocol
        inet addr:10.0.3.2  P-t-P:10.0.3.1  Mask:255.255.255.255
        UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:150  Metric:1
        RX packets:1 errors:0 dropped:0 overruns:0 frame:0
        TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0
        RX bytes:88 (88.0 b)  TX bytes:168 (168.0 b)

vegard@gyversalen:~$ ping -i 900 10.0.3.1[12]
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=6388671.9 ms

--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss
round-trip min/avg/max = 3211900.8/5222806.6/6388671.9 ms
vegard@gyversalen:~$ exit

Script done on Sat Apr 28 14:14:28 2001
```

有人可能会说这些人都是时间太多闲的，但正是这种创造性和热情才让开放源代码的运行（以及 Linux 在其中起到的作用）变得如此威力无穷和趣味无穷。Linux 式的摇滚。

30.16 主流的 Linux

Linux 目前是成长最快的操作系统。从大型机到手表的各种东西上都能找到它的身影。不过，它开始遇到一些在兼容性和微小差异上的问题，在 UNIX 的全盛时代这样的问题阻碍了 UNIX 的发展。

过去常在企业界听到这样的话，“哦，我们不能用 Linux——它得不到支持！”。但是世界变得如此之快，人们已经担心起微软来了。Linux 正在悄悄地渗透到商业领域。它常常先是由新聘用的学生们带来的，这些学生已经在大学里用过 Linux，并且在他们的台式机上公开地或是偷偷地运行过 Linux。在建立起了滩头阵地之后，Linux 经常变成了邮件或者 Web 服务器的首选平台，在这些地方，它的安全性、高性能和可扩展性让它比微软的专有解决方案更有优势。成本的低廉也大大给 Linux 加分。

“free software”中的“free”有两个意思：免费啤酒中的“免费”——您不用花钱——和“您在用的时候几乎不需要遵守什么限制”那样的“自由”。GPL（GNU Public License，GNU 公共许可证）落实了这两种含义，而这样做吓住了一些企业的 IT 部门。以采用 GPL 许可证的工作为基础取得的成果，如果要发布出来的话，必须继续遵守 GPL 的规定。有些管理者认为 Linux 开发人员就是长发的嬉皮士，用害怕、不可靠和怀疑的态度来对待 GPL。但是 Linux 似乎已经把他们争取过来了，虽然是慢慢争取的，但的确争取过来了。

过去几年里，www.distrowatch.com 已经根据其 Web 网站访问者的情况排出了前 100 名的 Linux 发行版本。我们所举的 5 个发行版本的相对位置有些变化。表 30.3 给出了一些数字。

[12] 注意，正确使用 RFC1918 中定义的私有地址空间。显然，需要基于信鸽的 NAT 实现来把测试网络同外界连接起来。

www.TopSage.com

表 30.3 过去几年 distrowatch.com 的流行度排行（截至 2006 年 9 月）

发行版本	过去 6 个月	过去 12 个月	2005	2004	2003
Ubuntu	1	1	1	13	-
SUSE	2	2	3	4	7
Fedora	3	4	4	2	-
Debian	7	7	6	5	5
Red Hat	29	24	23	12	2

每天点击的实际数字一般从少于 100 到超过 3 000 都有。更重要的是，这些排名清楚地显示出 Ubuntu 的增长，以及 RHEL 和 Fedora 的分裂。

有的大企业现在正在提供和支持 Linux 的开发：举几家公司的名字，有 IBM、HP、Compaq、SGI 和 Sun。有意思的事情是，能观察到在这些行动迟缓的软件巨头和由经验不足的年轻软件工程师所推动的 Linux 文化之间有了联姻，而这些年轻人的精力往往足以弥补他们的浮躁。IBM 已经在实施一次重大的变迁，以适应 Linux 开放源代码的无政府主义世界里的生活。IBM 已经能通过移植其经过良好测试的扩展代码库来做出自己的贡献。参考下面的站点了解更多的信息：

ibm.com/developerworks/opensource

SGI 也通过建立 Linux 测试计划（Linux Test Project）来对 Linux 做出贡献，这个项目是一系列测试套件和压力测试方法，能够证实 Linux 的可靠性、健壮性和稳定性。现在由 IBM 维护。最新的发布（2006 年 6 月）包括 2 900 多项测试，还有文档和测试工具。

LDP（Linux Documentation Project，Linux 建档计划）各级专业技术水平的文档，包括针对各种常见问题的稍长的介绍性小册子、FAQ、HOWTO 以及所有 Linux 核心命令和软件包的手册页[13]。LDP 还支持两本网上杂志：*Linux Gazette* 和 *Linux Focus*。所有文档都能通过 Internet 无偿获得。

至少有一家计算机入侵保险公司在他们的定价中认可了 Linux 的健壮性，Linux Web 服务器的保险费要比微软系统的等价产品少 5%～15%。在 2001 年 8 月，ZDNet 上的文章做出如下评价。

“保险经纪人承保经理 J.S. Wurzler 先生已经开始对使用 Microsoft IIS 软件的客户机收取最高可达 15% 的额外保费，因为它是 Code Red 蠕虫的美味佳肴……Wurzler 从 1998 年起就开始推销黑客保险了，这是根据他的公司在过去 3 年中所做的 400 多宗安全分析结果做出的决定。Wurzler 发现在开放源代码系统上工作的系统管理员似乎更聪明，任职时间比那些在使用 Windows 软件的公司里的系统管理员更长。”

在我们所举的 5 种发行版本中，两种得到了大型商业公司的支持：Red Hat 由 Red Hat 公司支持，而 SUSE 由 Novell 公司支持。Ubuntu 由南非人 Mark Shuttleworth 创办的一家小公司 Canonical Ltd 支持。Debian 和 Fedora 现在则是小家伙。

在 Linux 界，Debian 和更面向企业的发行版本（如 Red Hat 和 SUSE）之间已经形成了某种分歧。Debian 的人遇到了困难，他们的好主意没有得到别人的严肃对待，也没有融合到 LSB 的标准化工作中。我们希望最近流行的 Ubuntu 能有助于把 Debian 更牢固地保持在主流 Linux 的行列里。

CentOS（Community ENTerprise Operating System）是个英格兰的发行版本，它在想用稳定的 Linux 系统但又不买 Red Hat 的环境中流传着。CentOS 实质上就是 RHEL 的克隆。GPL 要求 Red Hat 公开它的软件补丁。CentOS 的人就给源代码树打上补丁，去掉 Red Hat 的徽标（logo）和商标，从头开始重新构建发行版本，做出的结果类似于 RHEL，但又不必花钱，也没有支持合同。当然，支持确实重要，RHEL 有 7 年的支持保证——接近 Linux 世界里的“永久”支持了。

对于大多数公司来说，软件所有权的总体成本也很重要。大多数 Linux 发行版本一开始都是免费

[13] 这个网站以前是 linuxdoc.org，但是由于一起内部丑闻：一名多心或者恶意的 Web 管理员声称自己享有这个名字的所有权，而且模仿办了一个叫做 linuxdoc.com 的网站，从而促使该项目改变了自己的域名。

www.TopSage.com

的，而且这些发行版本自身就可以通过网络进行管理，所以维护费用也便宜得多。Windows XP Professional 目前的价格是 250 美元/份，这还仅仅只是操作系统本身。一旦要加应用软件，对于大公司的价格就让人咂舌了。Google.com 目前运行着成百上千台 Linux 主机，设想一下，假如给那些机器每台都买 Windows 的许可证得要多少钱！

有几种 Linux 发行版本支持中文。Red Flag Linux 就是其中最著名的一种，据称它拥有 2 亿用户。

包括 NASA、NERSC、NIH、NOAA 和 USGS 这样的机构在内，有许多美国政府的部门都在使用 Linux。Fermilab、Los Alamos、Oak Ridge 和 Sandia 国家实验室也都在使用 Linux。如今最强大的计算机仍然是大型机系统——但是由 Linux 机器组成的 Beowulf 集群正在迎头赶上。在 2004 年 9 月，IBM 宣布它制造了当时世界上最快的超级计算机，一台在仅 8 个机架空间里安装 16 250 个 CPU 的计算机集群，它运行的 Linux 系统达到了每秒 36 万亿次浮点计算（teraflop）的稳定性能[14]。（译者注：NASA、NERSC、NIH、NOAA 和 USGS 分别为美国国家航空和航天局、美国国家能源研究科学计算中心、美国国立卫生研究院、美国国家海洋与大气管理局、美国国家地质调查局。Fermilab、Los Alamos、Oak Ridge 和 Sandia 国家实验室分别为美国国家加速器实验室[费米实验室]、美国洛斯阿拉莫斯国家实验室[美国三大实验室之一]、美国橡树岭国家实验室、美国桑迪亚国家实验室[美国三大实验室之一]。）

30.17 组织、会议及其他资源

许多 UNIX 和 Linux 的支持组织——既有一般性的，也有专门针对厂商的——能帮助您和其他使用同一软件的人们彼此进行沟通和交流。表 30.4 给出了这些机构的一个简明列表。还有大量的全国性的和地区性的组织，但没有在该表中列入它们。

表 30.4 Linux 和 UNIX 的组织

名 称	URL	说 明
LPI	www.lpi.org	Linux 专业协会（Linux Professional Institute）
LI	www.li.org	Linux 国际（Linux International）
FSF	www.fsf.org	自由软件基金会，GNU 计划的发起人
OSDL	www.osdl.org	开放源代码开发实验室，面向开发人员
USENIX	www.usenix.org	UNIX 用户组，技术性相当强
SAGE	www.sage.org	和 USENIX 有联系的系统管理员协会（System Administrators Guild），每年召开 LISA 大会
LOPSA	www.lopsa.org	职业系统管理员联盟，从 USENIX/SAGE 分出的组织
SANS	www.sans.org	召开系统管理和安全大会；没有 SAGE 技术性强，注重培训
EUROPEN	www.europen.org	过去是各国用户组织的联合国（UN），但现在基本上瘫痪了；剩下来的有 NLUUG、DUUG、UKUUG、GUUG 及其他
AUUG	www.auug.org.au	澳大利亚 UNIX 用户组，包括计算技术的技术和管理两方面
SAGE-AU	www.sage-au.org.au	澳大利亚 SAGE，每年在 Oz（译者注：澳大利亚的简称）召开一次大会
SANE	www.sane.nl	系统管理和网络工程组，每年在北欧召开大会

尽管大多数会议的覆盖面都超出了 Linux 或者 UNIX 的范围，而且还包括了 Windows 的发展和事件，但有些机构召开了针对有关 Linux 专题的会议。

Linux 国际（Linux International）和 Linux 专业协会（Linux Professional Institute）都在以各种方式推进 Linux——LI 通过市场营销上的努力来把 Linux 带入商业界（而且资助一些开放源代码的开发工

[14] 一个 teraflop 是指一秒执行一万亿次浮点计算。

www.TopSage.com

作），LPI 通过它的 Linux 系统管理员认证来推广 Linux，我们在下一小节讨论。

LI 的 Web 站点有许多对于管理人员来说很不错的资料，管理人员往往在使用开放源代码的操作系统上犹豫不决。LI 在几次商业展会上——包括 LinuxWorld、Linux Expo 和 CeBit——主办了一个 Linux 展台。LI 也领导着一项国际化的项目，把世界上所有的字符集都引入 Linux。他们在夏季举办的实习活动让许多学生和 Linux 开发人员搭档，手把手地教学生，让学生们有了编写高质量代码的经验。

LI 现在放慢了自己的脚步，因为它的目标，即把 Linux 引入企业界，已经完成了。LI 可能在不久的将来转变为一个某种 Linux 用户组织。

FSF（Free Software Foundation，自由软件基金会）是 GNU（GNU's Not Unix，一个递归的缩写词）项目的发起人。FSF 名字里的“free”是“自由言论”的那个自由的意思，而不是“免费啤酒”里的免费的意思。FSF 也是 GPL（GNU Public License，GNU 公共许可证）的发源地，GPL 涵盖了大多数 Linux 软件。GPLv3 现在制订中，其草案在公开征求意见。

OSDL（Open Source Development Lab，开放源代码开发实验室）是一家非盈利性的组织，其目标是加速 Linux 在企业界内的部署。OSDL 得到了大部分 Linux 客户的支持，为 Linux 开发界提供测试和技术支持。

30.17.1 大会和商业展览

表 30.5 详细列出了 3 个每年召开的 Linux 开发大会。

表 30.5 Linux 开发大会

会 议	URL	会议期限	地 点
Linux Symposium	www.linuxsymposium.org	4 天	加拿大渥太华
linux.conf.au	linux.conf.au	5 天	澳大利亚
Linux Kongress	www.linux-kongress.org	4 天	德国

渥太华大会通常在 8 月召开，包括 3 个论文系列和一个平行的课程系列。澳大利亚开发大会提供课程、研讨会、专题和小型会议，其中有一个专门针对系统管理。大会通常在 1 月召开，地点在澳大利亚的不同城市。德国的会议 Linux Kongress 包括两天的课程和两天的技术报告，大部分都是英语。它在德国各地巡回召开，最常开的时间是 9 月。

Linux Weekly News（lwn.net）是一份专门针对 Linux 的网上杂志。Elizabeth Coolbaugh 和 Jonathan Corbet 在 1997 年创办了它。这份杂志靠订阅获得支持，成员能够看到所有的文章，而非会员则必须等上大约一周的时间。订阅的价格按照您所处的行业而有所不同。“贫穷的黑客”最便宜，每月 2.5 美元，之后是“职业的黑客”，每月 5 美元，以及“项目领导”，每月 10 美元。

USENIX 是 Linux、UNIX 和其他开放源代码操作系统的用户组织。它每年召开一次综合性的大会和几次（较小的）专门性会议或者研讨会。综合性大会有一个与之同时举行的活动，专门致力于开放系统，这些系统体现出了 Linux 和 BSD 界正在进行的 OS 开发的特色。

对于系统管理员来说，最重要的事件是每年秋天召开的 USENIX LISA（Large Installation System Administration）大会。伴随这些会议还经常会举行商业展览。

最近几年来，USENIX 还专门为 Linux 内核开发开办了自己的专题研讨会，为 Linux 界提供服务。只能受邀参加这个为期两天的会议。

SAGE，即 USENIX 的系统管理员行业协会（System Administrators' Guild），它是系统管理员的第一个国际组织。它通过举办会议和一些非正式的活动把系统管理工作当作一种行业来向前推进。详细内容可参考 www.sage.org。

2005 年，USENIX 和 SAGE 之间关系的破裂给 SAGE 的未来留下了疑虑。其结果是 SAGE 组织里一些老前辈们组织了一个独立的组织 LOPSA（League of Professional System Administrators，职业系

www.TopSage.com

统管理员联盟），www.lopsa.org。他们还没有举办会议，但可能会很快就办。SAGE 曾有一项系统管理的认证计划，但已经放弃了，希望 LOPSA 能再办起来。

30.17.2 LPI：Linux 专业协会

LPI 提供 Linux 系统管理认证。这也就是说，它管理着能够衡量系统管理员从事各种 Linux 工作的能力和知识的测试。Linux 认证有 3 种水平，每种都包括两部分的测试。迄今为止，测试采取多重选择题的形式，并且通过 Web 由 Vue Electronic Testing Service（www.vue.com/lpi）负责管理。

1 级认证似乎涵盖了 Linux 高级用户的命令和非常基本的系统管理工作。2 级认证要更深入一些，包括了网络技术。3 级认证虽然规定了，但尚未完全成形。糟糕的是，自从本书出上一版以来，3 级认证的状态一直没有变化。LPI 的 Web 站点（www.lpi.org）列出了每一级认证所要求的知识类型，而且提供了从每种测试中选出来的样题。

认证很重要，要做好很难。系统管理是一门实践性很强的科学（或者，它是一种艺术吗？）。多选题虽然易于管理和评级，但是却不能很好地评测解决问题的能力，而这是好的系统管理员的标志。我们希望 LPI 认证系列中的 3 级认证能包括动手试验的部分，就像是 Cisco 的 CCIE 认证，而不要像 Microsoft 的 MCSE 考试。

Red Hat 和 SUSE 都有自己的认证计划。Ubuntu 的认证目前建立在 LPI 的认证之上。

30.17.3 邮递列表和 Web 资源

系统管理员已经有大量邮递列表和 Web 搜索引擎可供使用。我们在第 1 章里列举了一些我们最喜欢的资源。此外，从 www.linux.org/docs/lists.html 还可以得到一份清单，其中包含更多与 Linux 相关的邮递列表。

对于特定于 Linux 的问题，在 Google 上针对 Linux 做搜索相当有效。

也有一些邮递列表着重某些特殊的发行版本。表 30.6 给出了一些我们举例的发行版本所特有的邮递列表。

表 30.6 特定于发行版本的邮递列表

发行版本	URL
Red Hat	www.redhat.com/mailing-lists
Fedora	www.redhat.com/mailing-lists
SUSE	www.suse.com/us/support/online_help/mailinglists/
Debian	www.debian.org/MailingLists/
Ubuntu	lists.ubuntu.com

Red Hat 有数以亿计的邮递列表，其内容包括 RHEL 也包括 Fedora 的多个发行版本，在 www.redhat.com/mailing-lists 里列出了邮递列表的名字以及每个邮递列表的简述。除了上表里给出的 URL 之外，SUSE 还在 www.suse.com/us/business/mailinglists.html 有邮递列表的信息。这些邮递列表的存档则可以在 lists.suse.com/archives 找到。

30.17.4 系统管理方面的调查

和 USENIX 联合会（USENIX Association）有关的 SAGE（System Administrators Guild，系统管理员行业协会），还有 SANS（System Administration，Networking，and Security Institute，系统管理、网

www.TopSage.com

络和安全协会）每年都提供主要关注系统管理员薪酬补偿的调查报告，但报告也包括其他方面的内容。从 www.sage.org/salsurv 和 www.sans.org/salary2005 可以看到这些调查的报告。SAGE 最近的调查结果（50～60 页）只能供会员使用，而老的调查则任何人都能看到。SANS 的那个 URL 只提供一份 5 页的 PDF 文件，是最新调查结果的总结。

其中一些统计数据有：

- 薪水数据：薪水中间值、提升历史、奖金和加班费；
- 系统平台的组成；
- 每周工作时间；
- 对系统管理员的市场需求；
- 经验水平、教育水平和工作年限；
- 系统管理员的地理、性别和年龄分布；
- 工作中最麻烦的内容和最享受的内容。

把您单位的情况同这些报告反映的业界的情况进行比较很有意思。不过，调查数据必须仔细解读。这些结果会受到局部条件、经济状况的影响。薪水、生活成本和税率随国家甚至地区的不同而有很大变化。

30.18 推荐读物

30.18.1 基础设施

TRAUGOTT, STEVE. "*Bootstrapping and Infrastructure.*" Boston: LISA 1998.

www.infrastructures.org/papers/bootstrap/bootstrap.html

SCHWEIKERT, DAVID. "*ISGTC: an alternative to ~bofh/bin.*" Amsterdam, Netherlands: SANE 2004. isg.ee.ethz.ch/publications/papers/isgtc-sane.pdf

OETIKER, TOBIAS. "*TemplateTree II: The Post-Installation Setup Tool.*" San Diego, CA: LISA 2001. isg.ee.ethz.ch/tools/tetre2/pub/tetre-lisa.pdf

BURGESS, MARK S. *Principles of Network and System Administration (2nd Edition).* Hoboken, NJ: Wiley, 2003.

本章介绍管理的章节里介绍的许多工具和概念都可以在 isg.ee.ethz.ch/tools 找到。

30.18.2 管理

LIMONCELLI, THOMAS A., AND CHRISTINE HOGAN. *The Practice of System and Network Administration.* Boston, MA: Addison-Wesley, 2001.

这本书有一个 Web 网站 www.everythingsysadmin.com，这个网站也很有意思。在 2007 年还要出一个新版本。

MACHIAVELLI, NICCOLÒ. *The Prince.* 1513.

LIMONCELLI, THOMAS A. *Time Management for System Administrators. Sebastopol,* CA: O'Reilly Media, 2005.

BROOKS, FREDERICK P., JR. *The Mythical Man-Month: Essays on Software Engineering.* Reading, MA: Addison-Wesley, 1995.

这本优秀的小册子最早于 1975 年出版，其大部分内容仍然是真理。

www.TopSage.com

大学网 TopSage.com

MCCONNELL, STEVE. *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last. Redmond*, WA: Microsoft Press, 1998.

这本书写得很生动，各方面的内容都不错。

Pressman, Roger S. *Software Engineering: A Practitioner's Approach (6th Edition)*. Boston, MA: McGraw-Hill, 2005.

SOMMERVILLE, IAN. *Software Engineering (8th Edition)*. New York, NY: Addison Wesley, 2006.

如果您要了解和 ITIL 规程和标准有关系的大量术语和管理名词的意思，那么 www.itil-toolkit.com 这个网站是个很好的开端。

30.18.3 政策和安全

RFC2196, *Site Security Handbook*，及其前任 RFC1244。内容非常不同，所以两个版本都值得看。

30.18.4 法律问题、专利和隐私

EFF（电子先锋基金会）的 Web 网站 eff.org 是一个能获得对隐私、加密和法律问题的最新报道的好地方。读起来总是很有意思。

www.groklaw.net 是一个了解与 IT 有关的法律新闻的好网站。法律问题都是用通俗易懂的语言而不是律师的行话来介绍和总结的。

30.18.5 一般的业界新闻

slashdot.org 是一个了解技术新闻的好资源，但是经常比安全邮递列表这样的资源慢一两天。

HARROW, JEFFREY R. *The Harrow Technology Report*. www.theharrowgroup.com

这个网站有一些和技术问题有关的有趣文章。它把新闻和编辑的评论内容结合在了一起。J. R. Harrow 是以前由 Compaq 出版但现在已经停刊的期刊 *The Rapidly Changing Face of Computing* 的作者。这个网站一直维护到 2005 年年底。

www.heise.de 是一个很好的新闻网站。这是个德语网站，但是 www.heise.de/english 提供了英文翻译。

www.theregister.co.uk 是一个很好的 IT 新闻网站。

30.19 习题

E30.1 您单位里反复要做的工作步骤都有什么？哪些做得不频繁，可每次都和以前做的步骤不一样？哪些有风险？

E30.2 您依赖哪些外部的提供商？您需要和有 B（备份）方案吗？解释有或者没有的原因。如果有，介绍一下 B 方案。

E30.3 简短面试几个内部客户，判断其对获得计算基础设施有什么期望。这些愿望没有矛盾吧？它们合理吗？它们和给系统管理员小组规定的目标不矛盾吧？

E30.4 在您的站点已经为系统管理工作建立了什么样的有组织的基础？指出仍然缺少哪些组成部分？

E30.5 您的一位同事明天离开去吃午饭，结果就再没有回来，但是您还不知道是哪一位。

www.TopSage.com

这可能会影响到哪些关键性的处理工作？您的单位该如何准备来顶上这个离开的员工的工作？为了避免中断服务，必须有什么文档？

E30.6 如果您接下来的 3 个月都不在，会发生什么样的事情？当您最后回来的时候，有多少个同事会恨您？下面两星期您可以怎样做来减少这类事件造成的不愉快？

★E30.7 您的老板命令您在当年年底前把系统管理的预算削减 30%。您能量化这一削减造成的后果吗？提供一份总结，让老板在掌握情况的条件下，就哪些服务要减少或者中断做出决定。

★E30.8 当前支持 Linux 的大公司有哪些？他们的兴趣和动机是什么？他们做的发行版本是哪种？

★E30.9 您正在磁盘崩溃之后做清理工作，而且注意到了在目录 **lost+found** 中有若干文件。在进一步调查它们的时候，您发现其中有些文件是两个学生之间发送的邮件消息，他们在系里的防火墙上开了个后门，在主数据服务器上保存 MP3 文件。您应该怎么办？有现成的规章或者制度涵盖这样的事件吗？

★★E30.10 评估您的站点为新用户、系统管理员、标准规程和紧急情况准备的本地文档情况如何。

★★E30.11 预测各种商业的和免费的 UNIX 和 Linux 变体在今后 5 年的发展。随着时间的推移，当前的开发和发行模式怎样才能继续维持下去？硬件厂商吸纳 Linux 会造成什么样的长期影响？区分服务器和台式机市场的不同。

www.TopSage.com

跋

本书（英文版）是在 Windows XP 上写成的。我们使用 Adobe FrameMaker 来排版，用其他几种 Adobe 的软件来作插图和成书。

Tyler Curtain 用钢笔和墨水为本书画了卡通插图。我们在一台桌面扫描仪上把它们扫入计算机，用 Photoshop 把它们处理干净，然后用 Adobe Streamline 把它们转成 PostScript 文件（Streamline 的功能现在已经集成到 Adobe Illustrator 里了）。

本书（英文版）的正文文字采用由 Robert Slimbach 设计的 Minion MM 字体（译者注：Multiple Master，针对 Adobe 的 Type1 字体的操作）。标题、表和插图采用由 Robert Slimbach 和 Carol Twombly 设定的 Myriad MM 字体。

本书（英文版）的“代码”的字体是 Peter Matthias Noordzij 所设计的 PMN Caecilia。我们徒劳地去找一种看上去像是 Courier，但又不会出现 Courier 的许多排版问题的等宽字体，没有能找到。最后采用了现在这种成比例的字体，并且使用制表位（tab）来对齐输出的各列。这种方法效果相当好，但是 Caecilia 没有排技术术语时需要的一些字符，而且它的斜体版本明显比它的罗马体版本细得多。做这件事前后缺乏好字体的事实情况，反映出在另一个业已饱和和商品化的字体市场里仍然有相当大的缺口。字体设计人员要注意！

我们目前采用的软件对于我们的工作来说效果不错，但我们不可能长时间继续用它。Adobe 不再销售或者采用现有的 MM 字体，支持这种格式的功能已经系统地从 Adobe 产品中去除了。Adobe 还好像表现出要放弃 FrameMaker 的意思。遗憾的是，没有别的文档制作软件具有细致的交叉引用、索引和在本书中使用的根据条件处理文字的功能。我们正在积极地寻找一种替代方法，能够给 FrameMaker 用户提供一条可靠的转移途径。

www.TopSage.com